POLITECNICO DI MILANO

Computer Science and Engineering

Requirements Analysis and Specifications Document

CodeKataBattle

Software Engineering 2 Project

Academic year 2023 - 2024

22 December 2023

Version 1.0

Authors:
Eliahu Itamar COHEN
Marco GERVATINI
Caterina MOTTI

Professor:
Elisabetta DI NITTO

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of CodeKataBattle is to provide a platform where students can improve their software development skills by training with peers through code kata battles, a concept inspired by the discipline of martial arts katas, where practitioners repeatedly refine their techniques. Educators can create battles in which teams of students can compete against each other by developing a solution following a test-first approach. At the end of each battle, every group is evaluated to create a competition rank that measures each student's performance. In this document, we will delve into the CodeKataBattle platform, exploring its various features, functionalities, and provide an in-depth analysis of its components.

1.1.1 Goals

Below are presented the goals of CodeKataBattle. Further description will be discussed in section 2.2.

G.1 Educators can create and manage a tournament.

G.2 Educators can create new badges within a tournament.

G.3 Educators can create a battle within a tournament to which they have access to.

G.4 Students can subscribe to a tournament within the specified deadline.

G.5 Students can enroll in a battle within their tournaments.

G.6 Students can form a team by invite.

G.7 Students can see their team score in a battle.

G.8 Users can see the list of ongoing tournaments.

G.9 Users can see all tournaments ranking.

G.10 Users can visualize collected badges of students by visiting their profile.

1.2 Scope

CodeKataBattle is an online platform that allows students to improve their development skills in a entertaining way.
Educators use the platform to create a code kata battle in which teams can compete against each other. A battle is a programming exercise that the students are expected to solve by following a test-first approach. Each battle belongs to a tournament, that is also created by an educator. The platform shows to the registered students a list of published tournament and each student can choose to enroll into one of them (before the registration deadline) and form a group by inviting other students.
When the registration deadline expires, the platform automatically creates a GitHub repository containing all necessary code and send the link to all students who enrolled. Then the students must push their code in the repository triggering the CKB platform that automatically updates the battle score of the team.
The score is a natural number between 1 and 100. It consist of both an automatic and manual evaluation. The automatic evaluation is based on:

- functional aspects, measured in terms of number of test cases passed.
- timeliness, measured in terms of time passed between the registration deadline and the last commit.
- quality level of the sources, extracted thorough static analysis tool that consider multiple aspects selected by the educator at battle creation.

When the submission deadline expires, the educators can manually change the score (if needed) and then the platform automatically updates the personal tournament score of each students. These information are then available for all subscribed users.
The CKB platform also include gamification badges, a reward that represent the achievements of individual students. Educators can create a badge by specifying the title and one or more rules that must be fulfilled to achieve it. Each badge can be assigned to one or more students depending on the rules. Earned badges are visible in the personal profile of the student as well as battles won and tournaments ranking.

1.2.1 World Phenomena

WP.1 Students choose which tournament to participate to.

WP.2 Educators ideate the battle (problem's code, minimum/maximum number of student per group, registration deadline, final submission deadline, additional configuration for scoring).

WP.3 Students write a solution to the battle and push it in the GitHub repository.

1.2.2 Shared Phenomena

SP.1 Educators create a new tournament by inserting on the platform all needed information.

SP.2 Educators creates badges by inserting on the platform a title and one or more rules.

SP.3 Users visualize the list of published tournament.

SP.4 Students joins a new tournament.

SP.5 Students invites a team member or joins a team.

SP.6 Educators reviews the score manually for each student.

SP.7 Users visualize all students' personal profile with their collected badges.

SP.8 The system send a notification to all students when a new tournament is created.

SP.9 The system send a notification regarding new battles to all students enrolled in that tournament.

SP.10 The system send a notification when the final battle rank is available to all students that participated.

SP.11 The system send a notification when the final tournament rank is available to all students that participated.

1.3 Glossary

1.3.1 Definitions

Term	Definition
Users	Identify both students and educators that are logged in.
Notification	Sent via e-mail, from the system to the recipient.
Code Kata	A set of documents that consist in the description of the project (in which are listed all languages that are acceptable for a solution), the software project, test cases and build automation scripts.

1.3.2 Acronyms

Acronyms	Term
CKB	CodeKataBattle
G	Goal
WP	World Phenomena
SP	Shared Phenomena
R	Requirement
DA	Domain Assumption
SC	Scenario
UC	Use Case

1.4 Reference documents

- Project assignment specification document.
- Slides of software engineering 2 course on WeBeep.
- ISO/IEC 25010 standard.

1.5 Document Structure

This document is divided in 5 chapters, as follow:

1. **Introduction**: contains a summary of the given problem, focusing on all phenomena that must be taken into account and the goals that the system aim to achieve.
2. **Overall Description**: gives a general description of the system, focusing on its functions and constraints. It provides the domain assumptions of the analysed world.
3. **Specific Requirements**: explains in detail the functional and non functional requirements. It lists the possible interactions with the system in the form of scenarios, use cases and sequence diagrams.
4. **Formal Analysis Using Alloy**: contains a formal description of some critical aspects of the system by using Alloy.
5. **Effort Spent**: keeps track of the time spent to realize this document.
6. **References**: keeps track of the software used.

Chapter 2

Overall Description

2.1 Product perspective

2.1.1 Class diagram

In figure 2.1 is presented the high-level overview of the CKB system.
The main elements are:

- User: that can be either student or educator. It is identified by a unique username.
- Student: represent the personal profile of a student. It contains the earned badges, the number of battle won, and all tournament rankings.
- Team: identifies a group of students that can join a battle. Students must form a team in order to participate in a battle, even if it is formed by one.
- Tournament: identifies a tournament. It contain all useful information as well as all needed method(s) that allow the educators to create it and the students to enroll into it. It also contains the final ranking of all participants.
- Battle: identifies a battle. It contain all useful information as well as all needed method that allow the educators to create it and the team of students to join it. It also contains the ranking of all teams.
- Evaluator: it contains all files and methods that are necessary for both the automatic and manual evaluation.
- Badge: represent a badge with a title, description and a set of rules that regulates how a student can earn the badge based on some variables. It contains the method(s) that check whether a student can receive the badge.

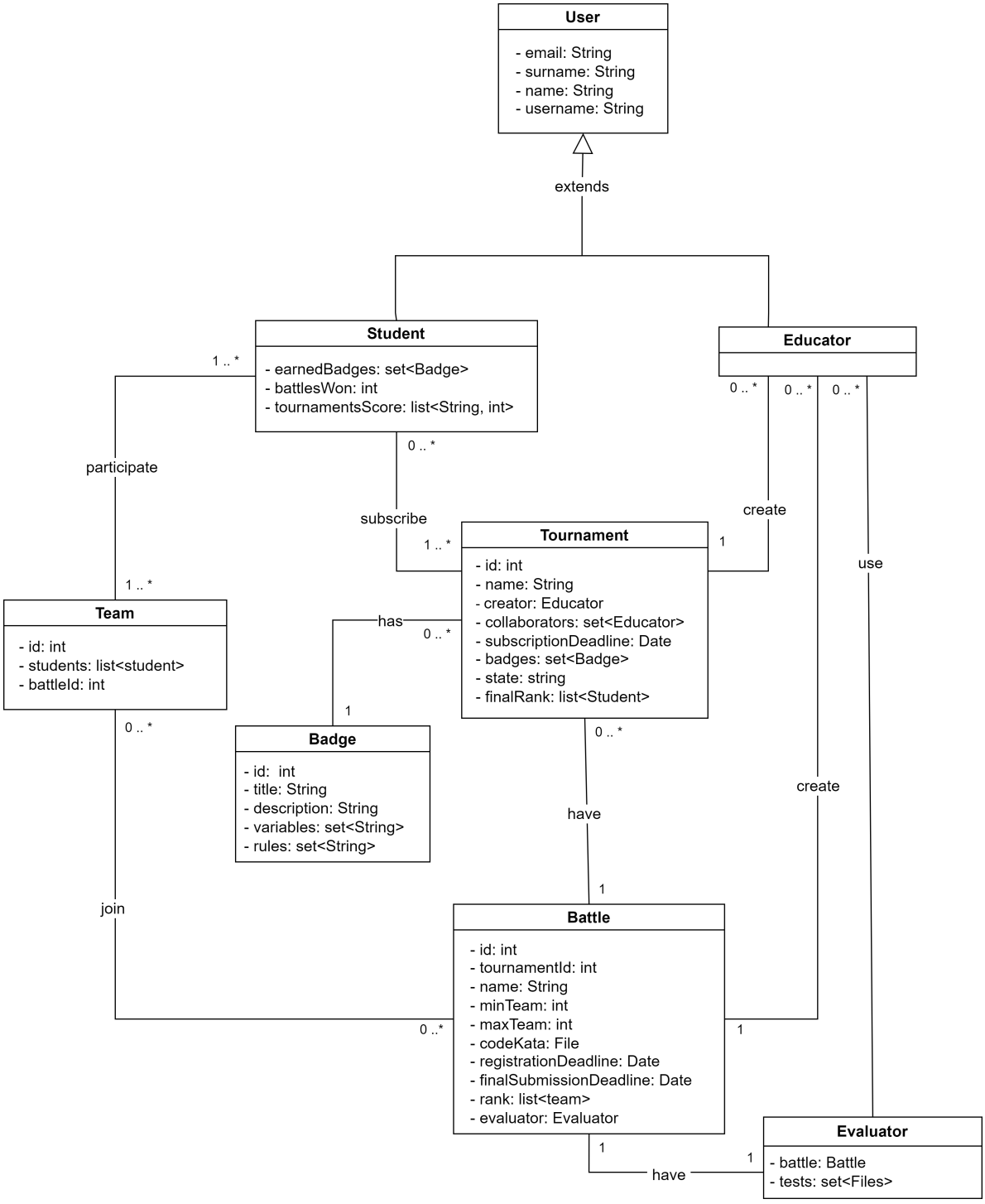

Figure 2.1: Domain class diagram

2.1.2 State diagrams

The following state diagrams describe the behaviour of the system focusing on the evolution over time of some particular aspects.

Tournament state diagram

The state diagram in figure 2.2 below represents the evolution of a tournament, from its creation to its closure.
As soon as an educator creates a new tournament with all necessary information, the enrollment stage opens and students can spontaneously register until the deadline expires.
While the tournament is ongoing, educators who have the permits can create new battles (further explanation below). It is possible that a tournament has no active battles, but it is still ongoing, for this reason we distinguished the two states clearly.
Only when all the battles are finished the creator can close the tournament, and the final ranking will then be available.

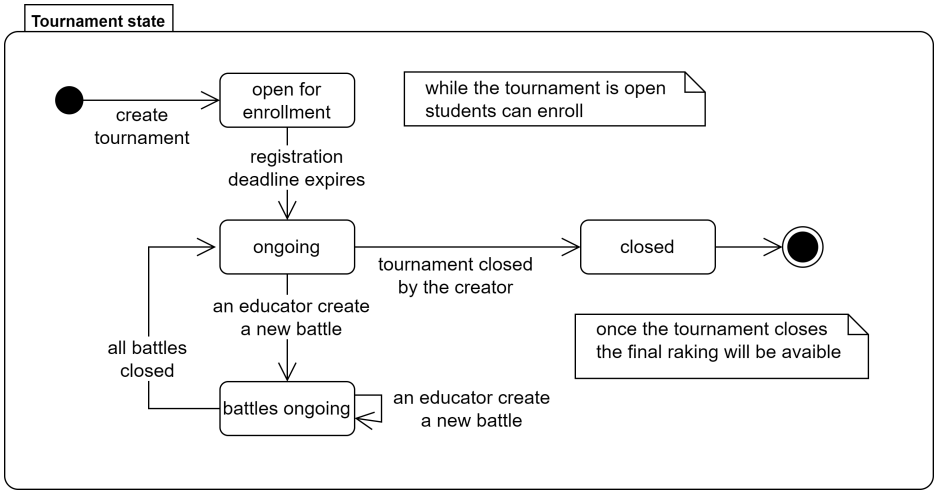

Figure 2.2: Tournament state diagram

Battle state diagrams

The state diagram in figure 2.3 below represents the evolution of a battle, from its creation to its end.
As soon as an educator create a new battle with all necessary information, the enrollment stage opens. Students, who are registered to the tournament in which this battle belongs, can enroll with a team (further explanation below).
When the registration deadline expires, the enrollment stage closes, the GitHub repository is automatically created and the battle is ongoing. As soon as a new push is available in a repository of a team the automatic evaluation is performed, and the ranking is updated.
When the final submission deadline expires the battle ends, a consolidation stage opens during which the educator can perform the manual evaluation (if needed).

When the educator finishes he/she can close the consolidation stage and the ranking will be available.

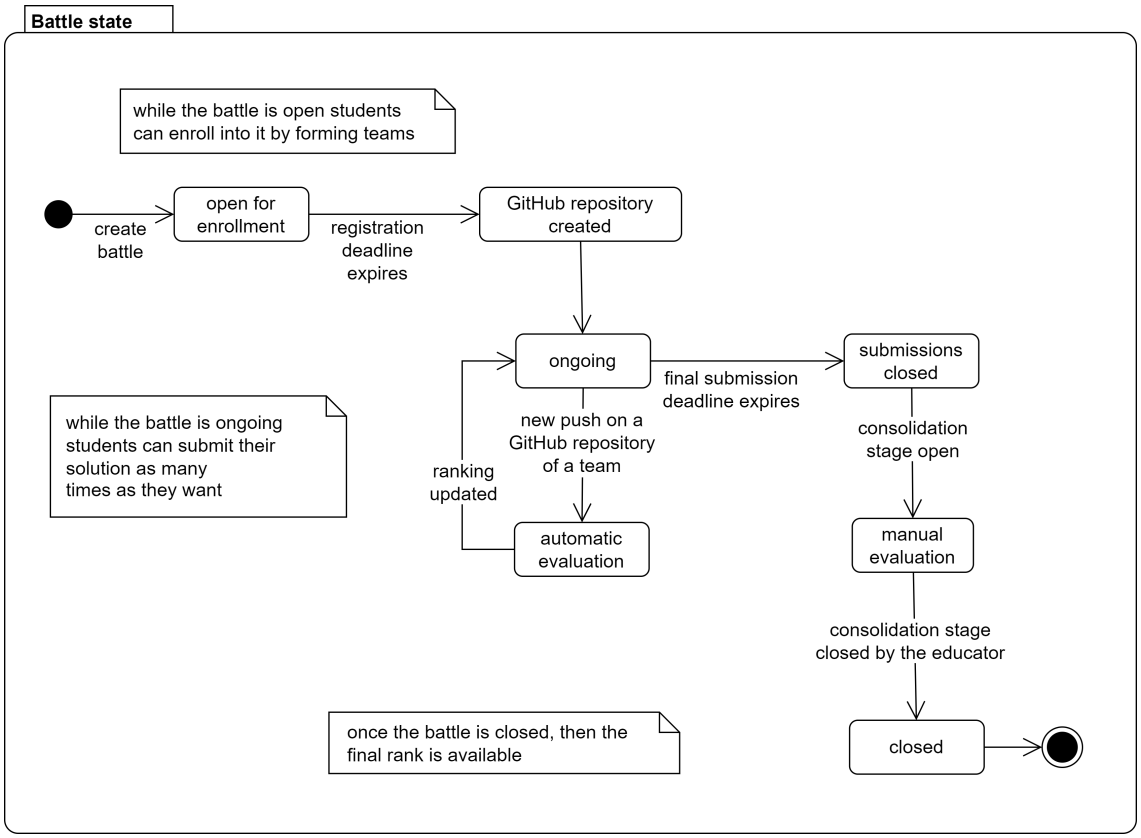

Figure 2.3: Battle state diagram

Team state diagrams

The state diagram in figure 2.4 below represents the evolution of a team.
A student can send infinite invitations, and the firsts to accept will be added to the team. As soon as the team reach the maximum number of students it is complete and all other invitations sent before will not be useful. A student can decide to leave the team at any time, but only before the registration deadline.
As soon as the battle registration deadline expires the team formation closes. If the team has reached the minimum required value it is complete and can participate to the battle. Otherwise it will be eliminated.

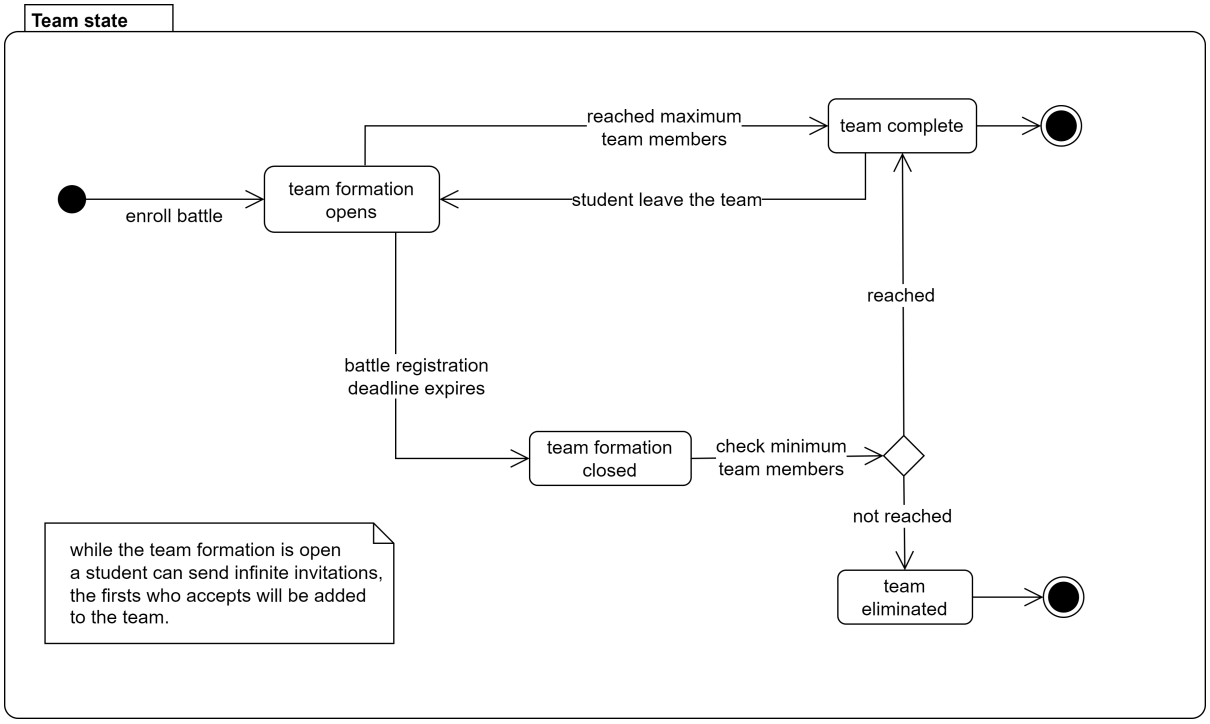

Figure 2.4: Team state diagram

2.1.3 Scenarios

In this section we will analyze all possible scenarios that can occur in using the CKB platform.

SC.1 Users sign up and log in.

Mario is a professor and wants to challenge his students, so he choose the CKB platform. He creates a personal account by inserting some personal information and then get a confirmation e-mail about the creation of a new account. He now can be able to create tournaments and battle.

Peach is a computer science student and wants to improve her software development skills. She learns about the CKB platform from her professor, and decide to create an account to be able to participate to the battles. The platform requests some personal information and then she gets a confirmation e-mail about the creation of her new account. She now have a personal profile where every other user can see her progress, in terms of badges earned, battles won, and tournament rankings.

In both cases the platform request some personal information such as name, surname, username, password, e-mail and the role, that can be student or educator. The account is created after the user clicks on the link sent in the confirmation e-mail. All future notifications will be sent through that e-mail.

Every registered user can log-in with username and password any time, in order to access the personal dashboard and use the CKB platform.

SC.2 Tournament and battle creation.

Mario, that is a registered educator, wants to create a new tournament called "Super Smash". He open the CKB platform, log-in, and from his personal dashboard he select the option "create new tournament". He then follows all the steps that allows him to set a title, a registration deadline and create badges. He can then invite other educators to create battles within his tournament. As soon as the tournament is created, an email is sent to all students registered to the platform, inviting them to subscribe.

Mario invites Luigi to his tournament by adding his username to the list of collaborators. Luigi will be notified by e-mail, and if interested have to join the tournament by clicking on the received link.

Luigi, that is also a registered educator, wants to create a new battle in Mario's tournament. He open the CKB platform, log-in, and from his personal dashboard he select "Super Smash" from the list of his tournaments. He then select the option "create new battle" and follows all the steps that allows him to upload the code kata, set a registration deadline, minimum and maximum number of students per group, a final submission deadline and selects relevant aspects that measures the quality level of the sources. The tournament "Super Smash" with the first battle is now open, in the list of all ongoing tournaments.

When all ongoing battles within the tournament have finished Mario can close it. Then the CKB platform will publish the final tournament ranking when is available and send a notification to all enrolled students.

SC.3 Students team formation.

Peach, Daisy and Toad, who are registered students, receive an e-mail that a

new tournament is open and they all decide to subscribe. They then receive another e-mail that a new battle is available and they decide to enroll.
Peach decides to form a team. She can invite other students enrolled in the battle by sending an invitation. She sends the invite to Daisy and Toad by specifying their username. They will receive a notification by e-mail, and if interested can join the team. Toad decided to join, while Daisy not.
If the team reach the minimum number of participants for the battle by the registration deadline, they can compete.
A team does not reach the minimum number of participants: Let's say that the minimum number of participant for the previously mentioned battle is 3, in this case Peach can send the invite to another student but she forgot and the registration deadline expires. Peach's team cannot compete in the battle, therefore their score will be automatically 0.

SC.4 Real-time battle progress.
Peach has formed a valid team for a battle within a tournament. The CKB platform has automatically created a GitHub repository with all the essential elements for the resolution of the battle, and send a link to all enrolled students. Peach and her team have to fork this repository and set up an automated workflow through GitHub Actions. At each pushed commit the platform automatically update the total score of the team. The team is also able to see the current ranking evolving during the battle, so they can improve their solution to get a better rank.
Students push the solution after the deadline: Every submission after the deadline is ignored.

SC.5 Manual evaluation.
After the final submission deadline of the battle expires, the consolidation stage open. During this phase Luigi, that is the creator of the battle, can optionally assign a personal score to each student that will be added to the total score of the entire team. As soon as Luigi has finished to evaluate all students, he will end the consolidation stage, and when the ranking are available all students will be notified by e-mail.
The score exceed the maximum value: Luigi can make a mistake and assign a score to a student making the overall team score higher than 100. In this case the score will be automatically cut off to 100.

SC.6 Gamification badges management.
Mario, that is a registered educator, at the creation of a tournament wants to add a badge that is earned by the student who wrote the most lines of code. From his personal dashboard he select the option "create a badge", then he has to specify the name, a description and one or more rules that regulates if a student is eligible to get that badge. In this case the name will be "Top Writer" and it will be awarded to the student who wrote the most lines of code within that tournament.
Peach, that is a student, aspires to earn the "Top Writer" badge, and so she heavily contributed on the tournament. At the end of the tournament the CKB platform automatically checks if any student is eligible for a badge, and Peach is assigned the "Top Writer", which will appear on her CKB profile.

SC.7 See student profile.

Mario, that is a registered educator, wants to check the profile of Peach, a registered student. In order to do this, he selects the option "see student profile" and write the username of Peach in the search bar. He will visualize Peach's profile, including all the badges that she has collected.

2.2 Product functions

In this section are explained the main functions that the CKB should provide to its users.

2.2.1 Sign Up and Login

These function will be available for all user.
All users can create a new profile by the sign up function. Each profile is uniquely identified by a username. Each user will be ask to provide the following personal information: name, surname, email, username, password and role (student or educator).
Login can occur every time using the username and password.

2.2.2 Create and manage a tournament

This function will be available only for educators.
Each educator can create a new tournament and invite other colleagues to create battles within that tournament. At the creation the educator must specify the title, the subscription deadline and eventual badges.
A tournament can then be closed by its creator, only when all battles within that tournament are closed.

2.2.3 Create gamification badges

This function will be available only to educators within the tournament creation.
A badge can be created by specifying a title, e description and at least one rule (which operates on the variables) that must be satisfied to achieve it. Each badge is then automatically assigned to one or more students by checking the validity of the rules at the end of the tournament. Educators can also create new variables that represent relevant information for scoring. A variable is identified by a unique name and a measurement unit, that can be selected by a list of predefined (e.g. integer, float, date, ..).

2.2.4 Create a battle within a tournament

This function will be available only for educators.
An educator can create a new battle in a tournament only if he/she created it or he/she have been invited to collaborate by another colleague. The system allow the educator to specify:

- code kata, that consist in the description and software project including test cases and build automated scripts.
- minimum and maximum number of students per group.
- registration deadline.
- final submission deadline.

- relevant aspects used by static analysis tools that measures the quality level of the sources.

When the registration deadline expires, the CKB platform creates a GitHub repository that contain the code kata, and sends the link to all the students enrolled in the battle.

2.2.5 Join a tournament

When a new tournament is created all registered students are notified by e-mail and can subscribe within a given deadline. By clicking on the link in the e-mail the student subscribe to the tournament.

2.2.6 Join a battle

When there is an upcoming battle in a tournament all subscribed students are notified by e-mail. By clicking on the link in the e-mail the student enrolls in the battle.

2.2.7 Create a team

This function will be available only for students who are enrolled in a battle.
Teams are formed by invite. The creator is able to send infinite invitations, and the firsts to accept will be added to the team.
The invite will be received by e-mail, and the recipient must click on the link if he/she wants to accept. The invite will expire in 48h.

2.2.8 Automatically evaluate

When the students push a new commit into the main branch of the GitHub repository, the CKB platform automatically analyze it and runs the tests to calculate and update the battle score of the team.
The score of each battle is evaluated in the following ways:

- functional aspects, measured in terms of number of test cases passed over all test available. The higher the better.
- time passed between the registration deadline and the last commit. The lower the better.
- quality level of the source, extracted through static analysis tools based on aspects chosen by the educator when creating the battle.

2.2.9 Manually evaluate

When a battle ends, a consolidation stage opens. The creator of the battle is able to optionally perform a manual evaluation for each student enrolled. Once the educator has evaluated all students he can end the consolidation stage, and when the final battle ranking is available all enrolled students will be notified by e-mail.

2.2.10 View all tournaments ranking

Every user can see the list of ongoing tournaments and the corresponding ranking (that compares all subscribed students' performance) in the CKB platform. At the end of each battle the personal tournament score of each student is automatically updated as the sum of all battle scores received within that tournament.

2.2.11 View my battles ranking

Every user that have access to a battle (in the case of an educator he/she is the creator, in the case of a student he/she is enrolled) can see the real-time progress of the ranking.

2.2.12 View a student profile

Every user can view a profile of a student by searching for his/her username. In each profile are shown badges earned, battles won, and tournaments ranking.

2.3 User characteristics

Each user must have a profile to be able to use the CKB platform. There are two different type of users:

- **Students**: The students are the primary users of the CKB platform. They range from beginners to advanced learners. Each student has a personal profile that display their badges earned, battles won, and tournament rankings.
- **Educators**: The educators can create and manage tournaments and/or battles. They challenge the students and then evaluate them based on their submissions. An educator must be officially recognized as he/she should have the adequate knowledge.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

In this section are listed all assumption made for the domain in which the system operates. These are conditions that the system take for granted because they are external to it but influence its behaviour.

DA.1 Users must have internet connection to interacts with the CKB platform.

DA.2 Students must have a GitHub account to be able to deliver a solution.

DA.3 Each team must fork the GitHub repository created by the CKB platform once and set up an automated workflow through GitHub Actions.

DA.4 The automatic workflow correctly trigger the CKB platform in order to evaluate the new push.

DA.5 Students must have a development environment with the necessary software tools and libraries to complete code kata battles.

DA.6 Educators must be qualified and experienced in software development and related fields.

DA.7 The uploaded "code kata" by the educators must be correct and complete.

DA.8 Educators must create badges that are not redundant with other ones in the tournament.

DA.9 The static analysis tools used must be reliable, and perform correct evaluations

Chapter 3

Specific Requirements

3.1 External interface Requirements

3.1.1 User Interfaces

The images below will present an idea of the user interfaces of the major function of the system. Since educators and students uses the platform differently, they will have access to two different personalized dashboard.
Button coloured in yellow and represents the main function that the CKB platform offers. The green fields represent area where the user must enter information (forms-like).

The figure 3.1 represent the front page to which each user will have access to, and from which he/she can decide to sign up or log in.

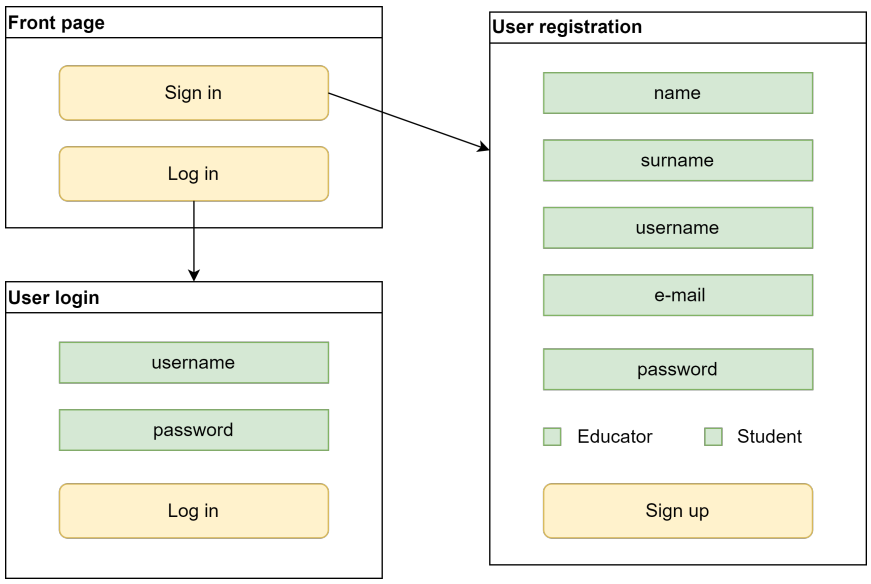

Figure 3.1: Start page with log in and sign up

Used by the educators.

The "All Tournaments" section is the same as the student dashboard in the next image.
The yellow buttons marked with (*) are those that are not always available. For instance in "Manage tournament" the "Close tournament" button is available only when all battles within that tournament are closed. Furthermore, in "Manage battle" the "Perform manual evaluation" button is available only after the final submission deadline of the battle expires and the "Close consolidation stage" button is available only when the educator has performed the manual evaluation.

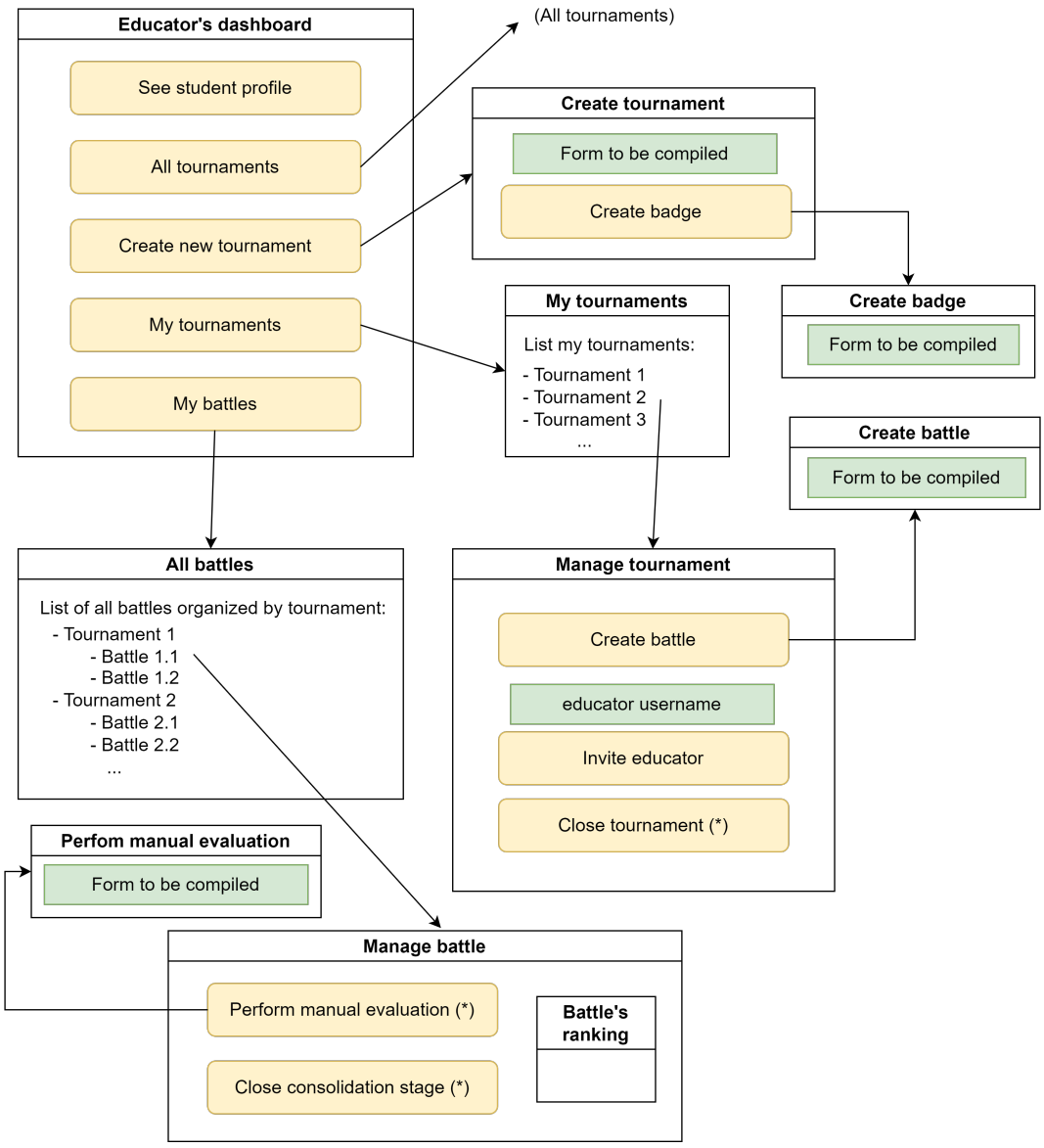

Figure 3.2: Educator dashboard

Used by the students.

As stated before yellow buttons marked with (*) are those that are not always available. For instance in "Battle dashboard" both the "Invite student" and "Leave team" buttons are available only when the registration deadline for that battle has not yet expired.

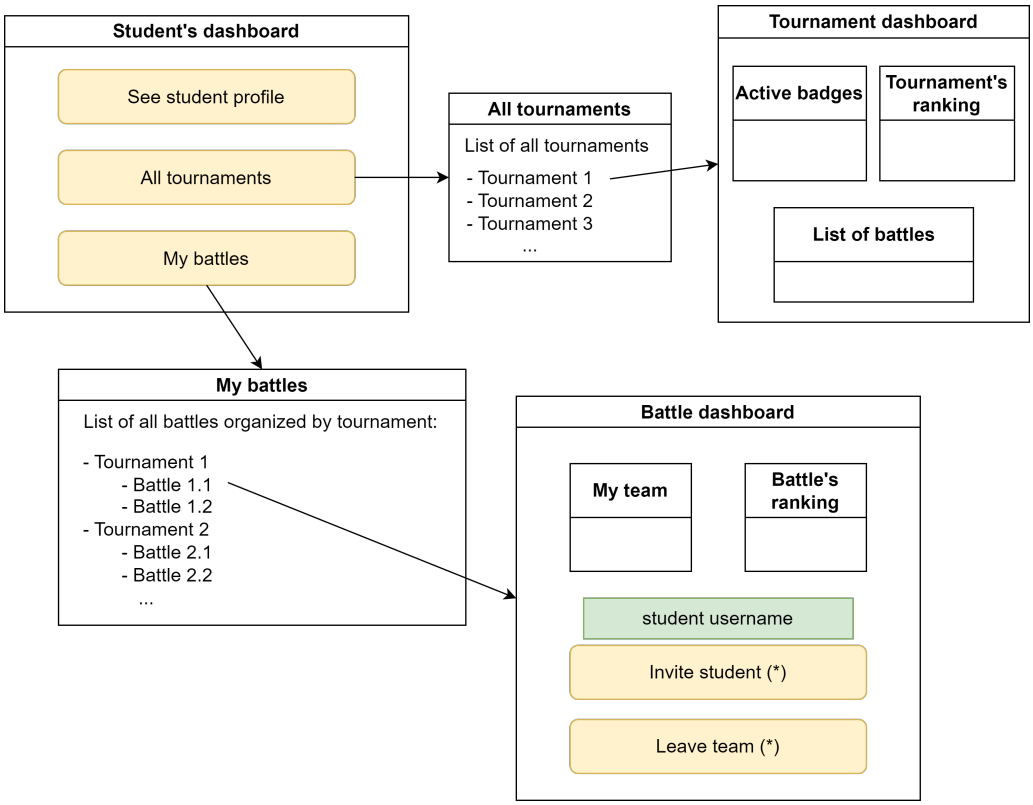

Figure 3.3: Student dashboard

3.1.2 Hardware Interfaces

The only hardware interface required is the personal device of the user (computer, tablet or smartphone) that will access the CKB platform through the web browser.

3.1.3 Software Interfaces

The CKB systems uses the GitHub API in order to detects a new push to the main branch of each forked repository. This will trigger the automated workflow that will evaluate the current submission.
A static analysis tool is also used in order to extract quality level of the sources based on parameters selected by the educators in battle creation.

3.1.4 Communication Interfaces

The only communication interface used is internet, via HTTP.

3.2 Functional Requirements

In this section, it is given a complete description of the functional requirements of the system.

3.2.1 Requirements

Users

R.1 The system shall allow users to register on the platform with their personal information.

R.2 The system shall allow registered users to log in the platform with valid credentials.

R.3 The system shall allow all users to see the list of ongoing tournaments.

R.4 The system shall automatically evaluate submissions.

R.5 The system shall allow users to be able to monitor ranking in real-time during battles and tournaments.

R.6 The system shall allow users to see other students' profile.

Educators

R.7 The system shall allow the educators to create tournaments.

R.8 The system shall allow the educators to manage their tournaments, in particular invites other collaborators and ends the tournament.

R.9 The system shall allow the educators to create code kata battles within a tournament.

R.10 The system shall allow educators to manually evaluate students (if needed) right after a battle closes, during the consolidation stage.

R.11 The system shall allow educators to create badges.

R.12 The system shall allow educators to define rules and variables for a badge.

Students

R.13 The system shall allow students to subscribe in tournaments.

R.14 The system shall allow students to enroll in a battle within a tournament.

R.15 The system shall allow students to form team for battles, by inviting other students.

R.16 The system shall notify the students about new tournaments within 10 seconds.

R.17 The system shall notify the students about upcoming battles in tournaments in which they are subscribed within 10 seconds.

R.18 The system shall notify the students when they receive an invite to participate in a team within 10 seconds.

R.19 The system shall notify the students when the GitHub repository of a battle is available.

R.20 The system shall notify the students when the final battle rank became available within 10 seconds.

R.21 The system shall notify the students when the final tournament rank became available within 10 seconds.

3.2.2 Goal mapping on requirements and domain assumption

G.1 Educators can create and manage a tournament.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.7** The system shall allow the educators to create tournaments. **R.8** The system shall allow the educators to manage their tournaments, in particular invites other collaborators and ends the tournament. **R.16** The system shall notify the students about new tournaments within 10 seconds.
DA.1 Users must have internet connection to interacts with the CKB platform. **DA.6** Educators must be qualified and experienced in software development and related fields.

G.3 Educators can create a battle within a tournament to which they have access to.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.9** The system shall allow the educators to create code kata battles within a tournament. **R.17** The system shall notify the students about upcoming battles in tournaments in which they are subscribed within 10 seconds.
DA.1 Users must have internet connection to interacts with the CKB platform. **DA.7** The uploaded "code kata" by the educators must be correct and complete. **DA.6** Educators must be qualified and experienced in software development and related fields.

G.4 Students can subscribe to a tournament within the specified deadline.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.3** The system shall allow all users to see the list of ongoing tournaments. **R.13** The system shall allow students to subscribe in tournaments. **R.16** The system shall notify the students about new tournaments within 10 seconds.
DA.1 Users must have internet connection to interacts with the CKB platform.

G.5 Students can enroll in a battle within their tournaments.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.14** The system shall allow students to enroll in a battle within a tournament. **R.17** The system shall notify the students about upcoming battles in tournaments in which they are subscribed within 10 seconds. **R.19** The system shall notify the students when the GitHub repository of a battle is available.
DA.1 Users must have internet connection to interacts with the CKB platform. **DA.2** Students must have a GitHub account to be able to deliver a solution. **DA.5** Students must have a development environment with the necessary software tools and libraries to complete code kata battles.

G.6 Students can form a team by invite.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.15** The system shall allow students to form team for battles, by inviting other students. **R.18** The system shall notify the students when they receive an invite to participate in a team within 10 seconds.
DA.2 Students must have a GitHub account to be able to deliver a solution. **DA.5** Students must have a development environment with the necessary software tools and libraries to complete code kata battles. **DA.3** Each team must fork the GitHub repository created by the CKB platform once and set up an automated workflow through GitHub Actions. **DA.1** Users must have internet connection to interacts with the CKB platform.

G.7 Students can see their team score in a battle.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.4** The system shall automatically evaluate submissions. **R.10** The system shall allow educators to manually evaluate students (if needed) right after a battle closes, during the consolidation stage. **R.5** The system shall allow users to be able to monitor ranking in real-time during battles and tournaments. **R.20** The system shall notify the students when the final battle rank became available within 10 seconds.
DA.1 Users must have internet connection to interacts with the CKB platform. **DA.4** The automatic workflow correctly trigger the CKB platform in order to evaluate the new push. **DA.9** The static analysis tools used must be reliable, and perform correct evaluations

G.8 Users can see the list of ongoing tournaments.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.3** The system shall allow all users to see the list of ongoing tournaments. **R.16** The system shall notify the students about new tournaments within 10 seconds.
DA.1 Users must have internet connection to interacts with the CKB platform.

G.9 Users can see all tournaments ranking.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.5** The system shall allow users to be able to monitor ranking in real-time during battles and tournaments. **R.21** The system shall notify the students when the final tournament rank became available within 10 seconds.
DA.1 Users must have internet connection to interacts with the CKB platform.

G.2 Educators can create new badges within a tournament.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.11** The system shall allow educators to create badges. **R.12** The system shall allow educators to define rules and variables for a badge.
DA.1 Users must have internet connection to interacts with the CKB platform. **DA.6** Educators must be qualified and experienced in software development and related fields. **DA.8** Educators must create badges that are not redundant with other ones in the tournament.

G.10 Users can visualize collected badges of students by visiting their profile.
R.1 The system shall allow users to register on the platform with their personal information. **R.2** The system shall allow registered users to log in the platform with valid credentials. **R.6** The system shall allow users to see other students' profile.
DA.1 Users must have internet connection to interacts with the CKB platform.

3.2.3 Use cases

In figure 3.4 is represented a general overview of all use cases deduced from the scenarios in section 2.1.3. All cases include "User login" because all function provided by the platform are accessible to registered users. (except for user sign up)

Figure 3.4: Use cases diagram

UC.1 User sign up

Name	User sign up.
Actors	Educators, Students.
Entry conditions	An user enters the CKB platform for the first time.
Event flow	1. The user visualize the sign up page. 2. The user inserts the requested personal information. 3. The user clicks on the "sign up" button. 4. The system checks if the username is available. 5. The system sends a confirmation e-mail to the user. 6. The user clicks on the link on the confirmation e-mail received. 7. The system creates the personal profile of the user.
Exit conditions	The user is registered in the system.
Exceptions	• If the chosen username is not available the system will throw an error message and the user will be requested to choose another one.

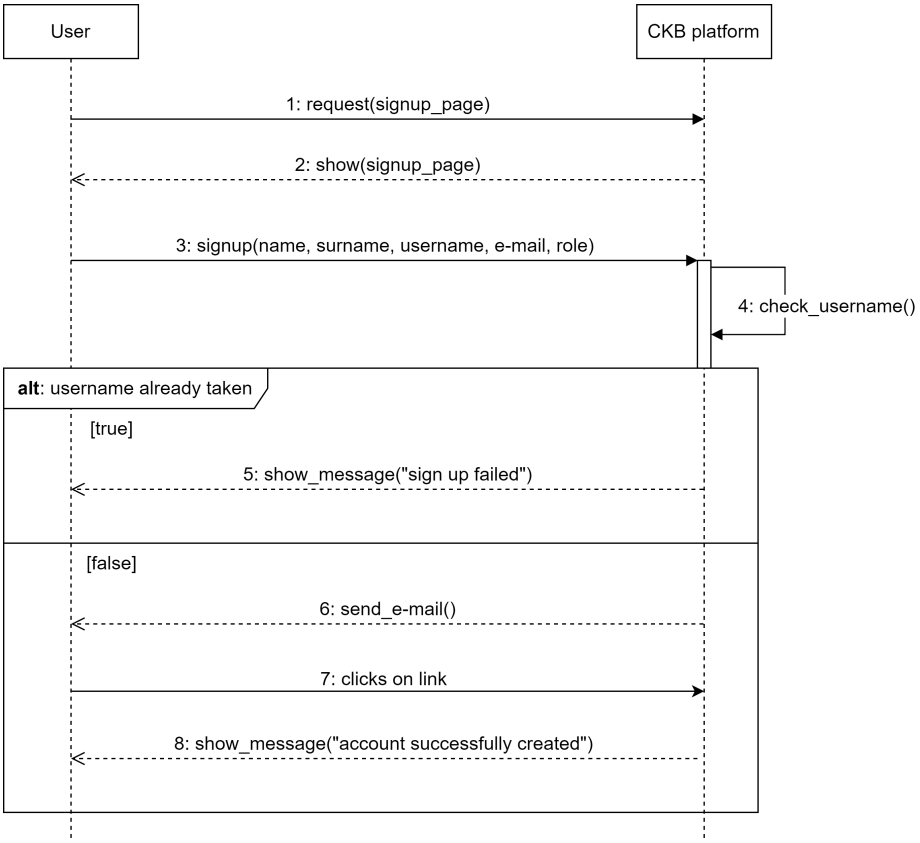

Figure 3.5: Sign up sequence diagram

UC.2 User login

Name	User login.
Actors	Educators, students.
Entry conditions	The user opens the CKB platform and clicks on "log in" button.
Event flow	1. The user visualize the log in page. 2. The user inserts username and password in the form. 3. The user clicks on "log in" button. 4. The system checks the credentials. 5. The system show the personal dashboard.
Exit conditions	The user has entered the CKB platform successfully.
Exceptions	If username and/or password are not correct the system will throw an error message and return to the entry condition.

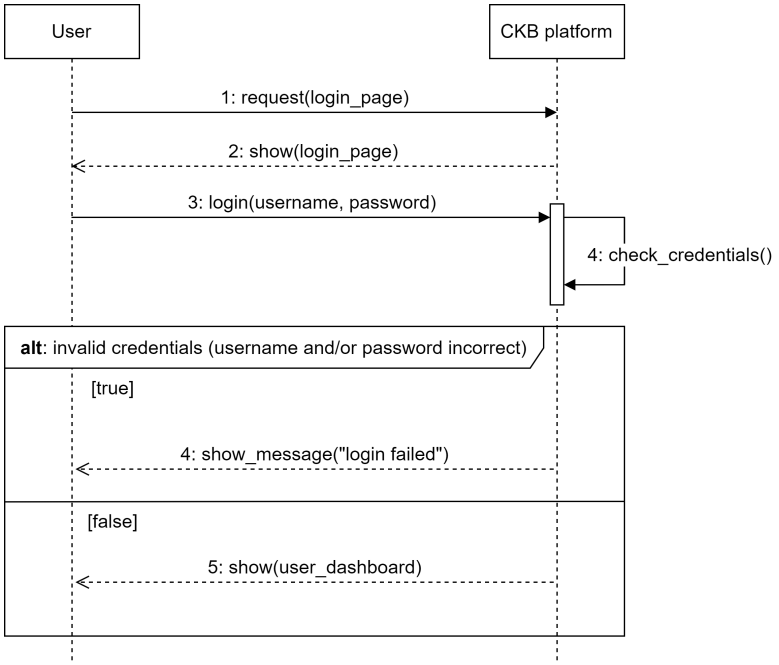

Figure 3.6: Log in sequence diagram

UC.3 Create a tournament

Name	Create a tournament.
Actors	Educators.
Entry conditions	An educator, who is logged in the platform, clicks on the button "create tournament".
Event flow	1. The educator inserts all needed information in the form. 2. The system checks that the correctness of all information.
Exit conditions	The tournament has been successfully created and can be visualize in the list of all ongoing tournaments. All registered students will receive a notification that a new tournament is available.
Exceptions	If there are some incorrect information, the system will throw an error message and the educator will be requested to modify it. The system return to the entry condition.

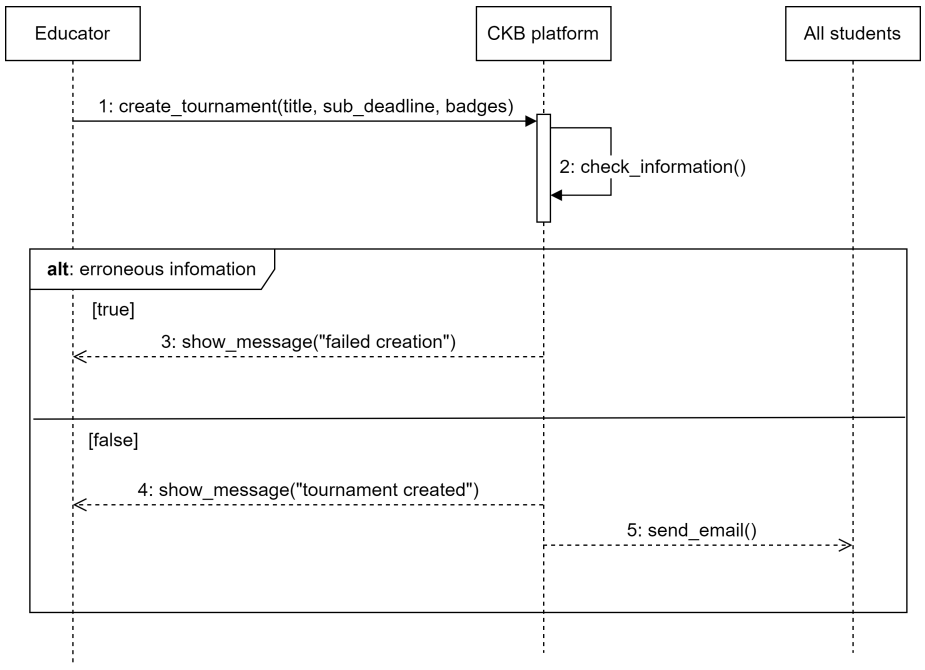

Figure 3.7: Create tournament sequence diagram

UC.4 Create battle

Name	Create a battle.
Actors	Educators.
Entry conditions	An educator, who is logged in the platform, wants to create a battle.
Event flow	1. The educator clicks on the button "my tournaments". 2. The educator visualize a list of tournaments to which he/she has access. 3. The educator select a tournament from the list. 4. The educator clicks on the button "create battle". 5. The educator inserts all needed information in the form. 6. The system checks that the correctness of all information.
Exit conditions	The battle has been successfully created and all students subscribed to the tournament in which it belongs have received a notification.
Exceptions	If there are some incorrect information, the system will throw an error message and the educator will be requested to modify it. The system return to the entry condition.

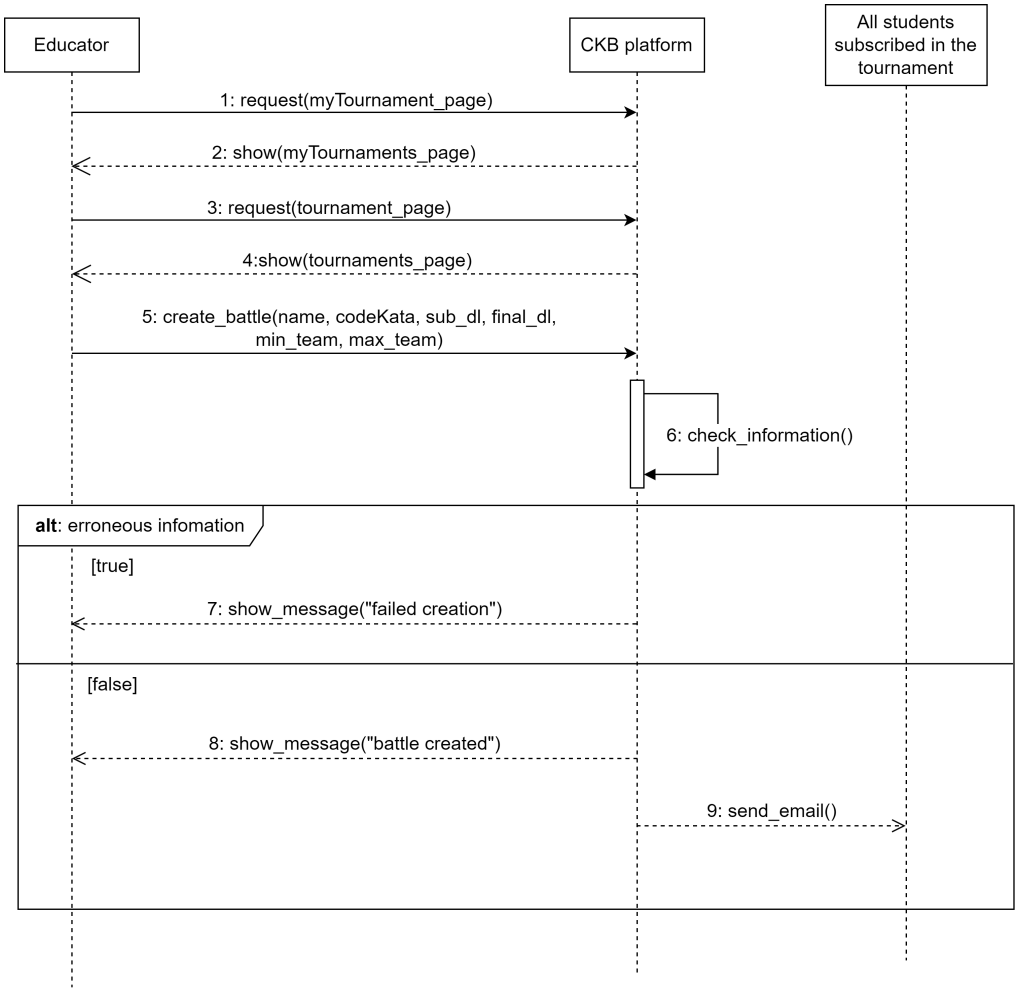

Figure 3.8: Create battle sequence diagram

UC.5 Invite an educator

Name	Invite an educator.
Actors	Educators.
Entry conditions	An educator, who created a tournament, wants to invite some colleagues to participate.
Event flow	1. The educator clicks on the button "my tournaments". 2. The educator visualize a list of tournaments to which he/she has access. 3. The educator select a tournament from the list. 4. The educator insert the username of the colleague he/she wants to invite. 5. The educator clicks on the button "invite educator" 6. The system sends an e-mail to the selected colleague.
Exit conditions	The invite has been successfully sent.
Exceptions	If the selected username does not exist or belongs to a student the system will throw an error message. The system will return to the entry conditions.

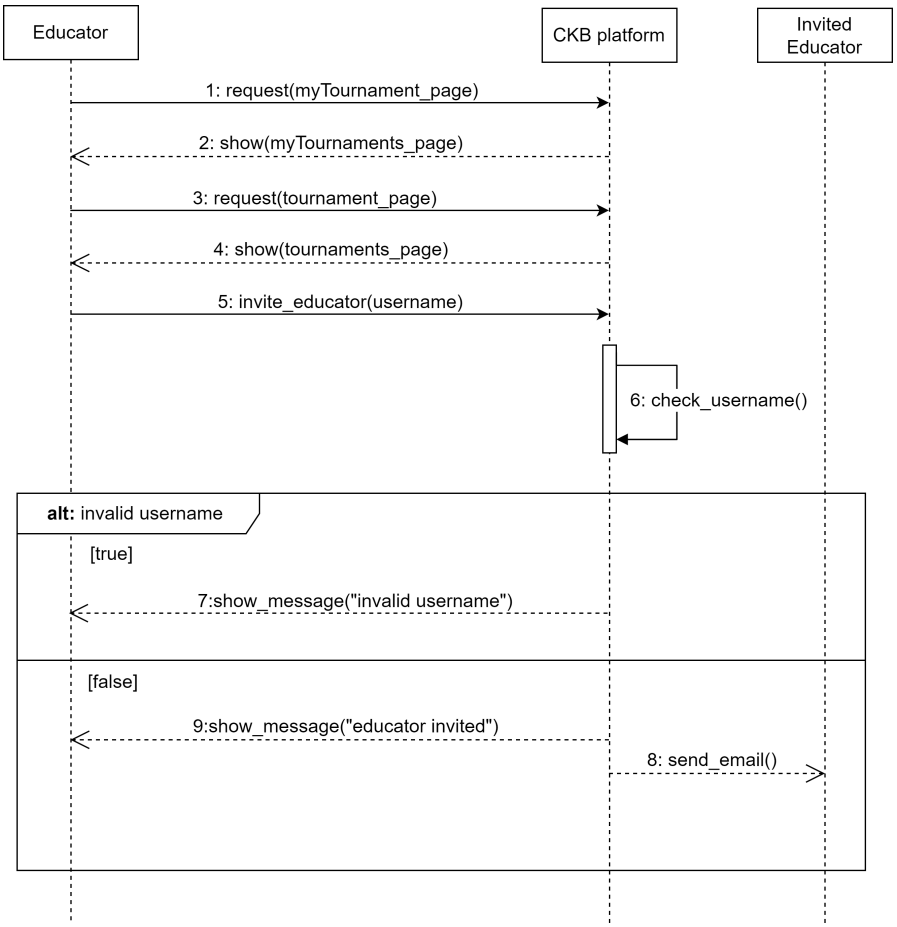

Figure 3.9: Invite educator sequence diagram

UC.6 Educator accepts invite

Name	Educator accepts an invite.
Actors	Educators.
Entry conditions	An educator receives an invitation by email and he/she wants to join the tournament.
Event flow	1. The educator clicks on the link received by email. 2. The system checks that the tournament is not closed.
Exit conditions	The educator has successfully joined the tournament.
Exceptions	If the tournament is closed the system will throw an error message and the educator will not be able to join it.

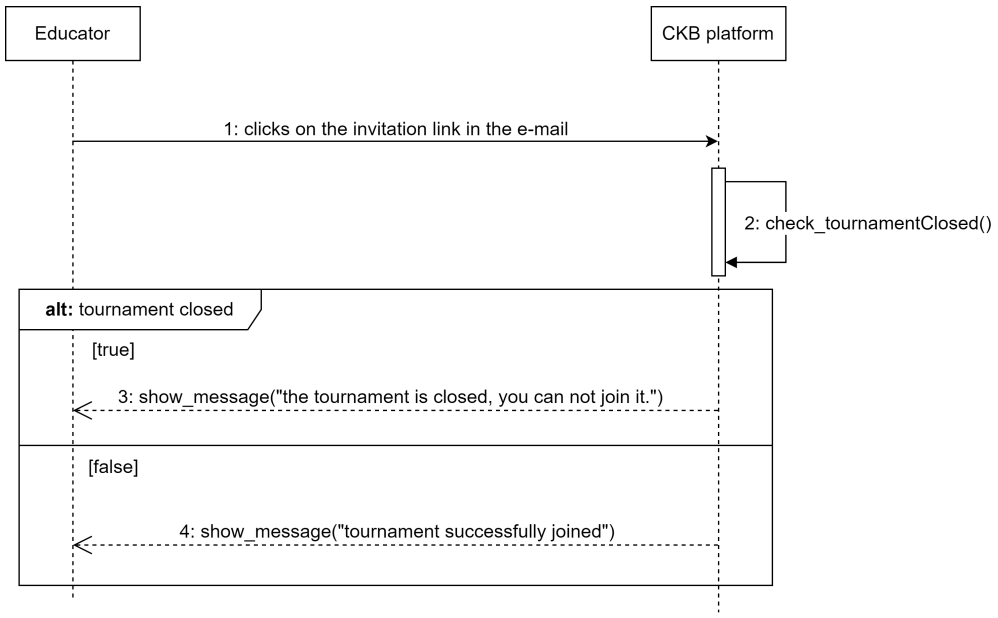

Figure 3.10: Educator accepts an invite sequence diagram

UC.7 Close a tournament

Name	Close a tournament.
Actors	Educators.
Entry conditions	An educator, who created a tournament, wants close it.
Event flow	1. The educator clicks on the button "my tournaments" 2. The educator visualize a list of tournaments to which he/she has access. 3. The educator select a tournament from the list. 4. The educator clicks on the button "close tournament". 5. The system checks whether all battles in that tournament are finished. 6. The system checks if any student have fulfilled some badges rules (if present). 7. The system produces the final ranking of the tournament.
Exit conditions	The tournament has been successfully closed. The badges have been assigned correctly. The final ranking is available. All students who participated are notified by e-mail.
Exceptions	If not all battles of the tournament are over, the system will throw an error message and the educator will not be able to close the tournament. The system returns to the entry condition.

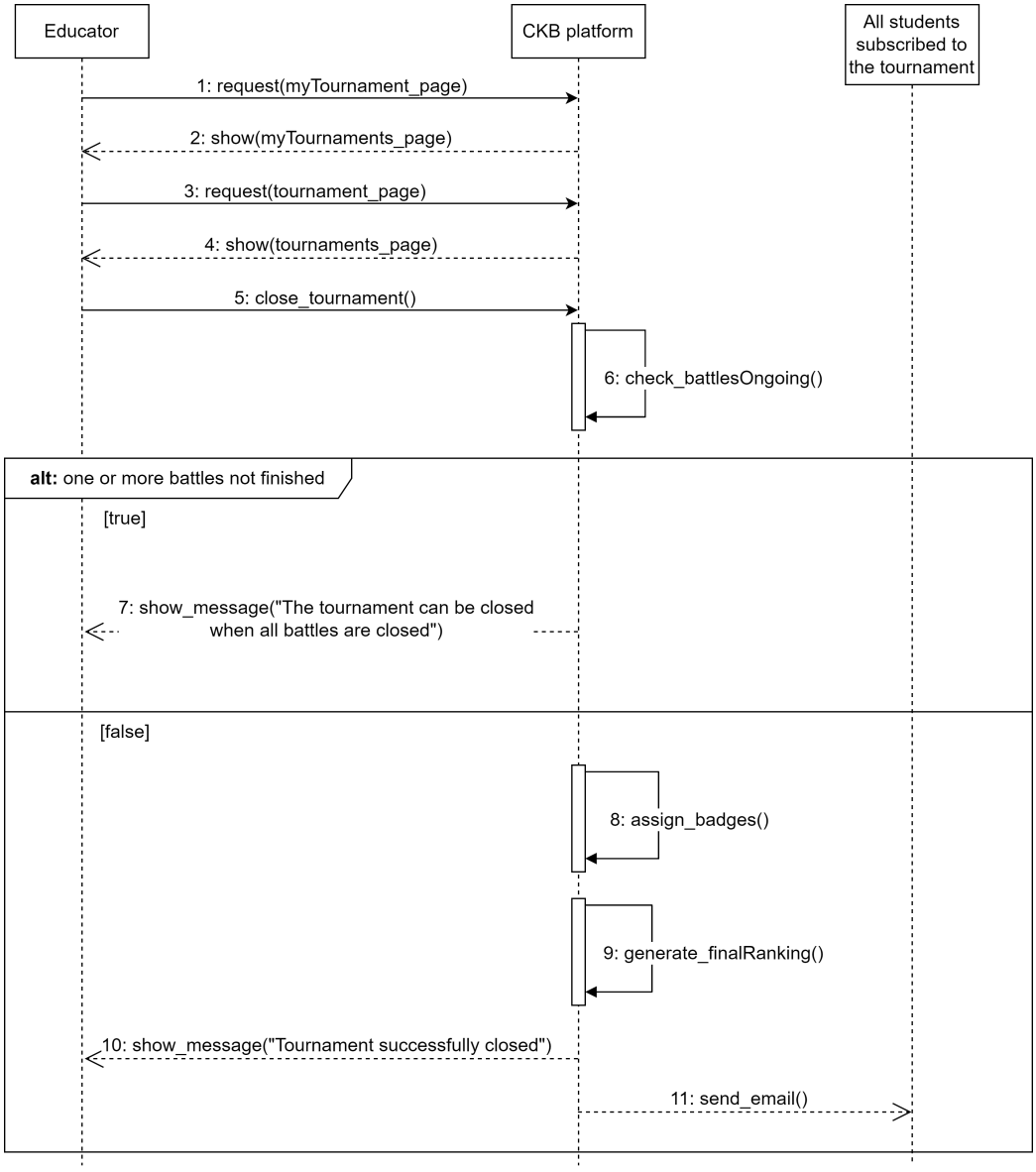

Figure 3.11: Close tournament sequence diagram

UC.8 Visualize all ongoing tournament with ranking

Name	Visualize all ongoing tournament with ranking.
Actors	Educators and students.
Entry conditions	A user wants to see the list of all tournaments and/or their rankings.
Event flow	1. The user clicks on the button "see all tournaments" 2. The user visualize a list of tournaments. 3. The user clicks on a tournament to visualize its ranking.
Exit conditions	The user visualize a list of tournaments and/or a specific ranking.

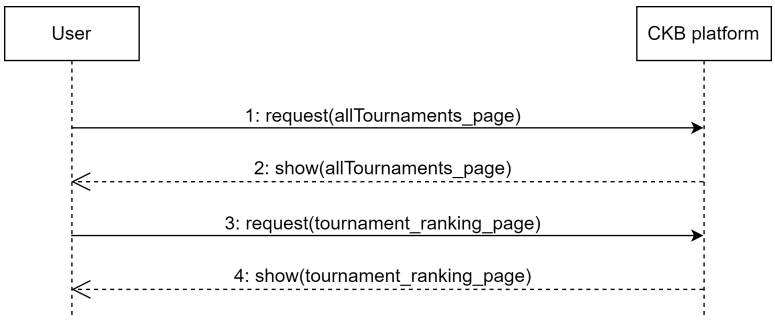

Figure 3.12: Visualize all tournaments ranking sequence diagram

UC.9 Subscribe to a tournament

Name	Subscribe to a tournament.
Actors	Students.
Entry conditions	A student, who is registered to the platform, receive a notification that a new tournament is available and wants to subscribe.
Event flow	1. The user clicks on the link received via e-mail. 2. The system check if the subscription deadline is not expired.
Exit conditions	The student is successfully subscribed to the tournament. The system will send the student further notification about the upcoming battles.
Exceptions	If the subscription deadline is expired, the system will throw an error message and the student will not be able to subscribe to the tournament. The system will return to the entry condition.

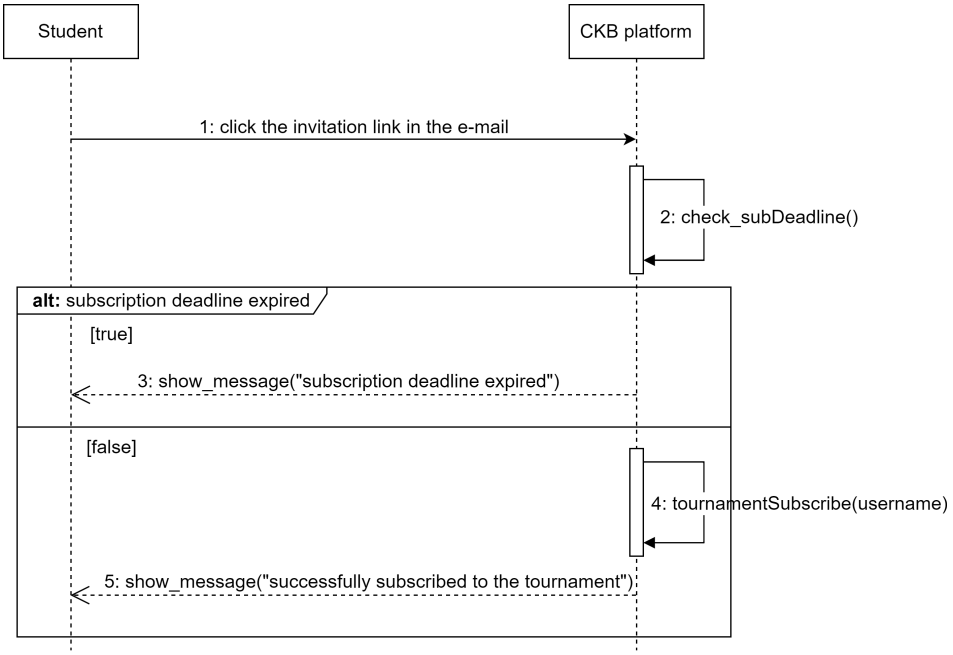

Figure 3.13: Subscribe to a tournament sequence diagram

UC.10 Enroll in a battle

Name	Enroll in a battle.
Actors	Students.
Entry conditions	A student, who is subscribed to a tournament, receive a notification that a new battle is available and wants to enroll.
Event flow	1. The student clicks on the link received by e-mail. 2. The system checks if the registration deadline is not expired.
Exit conditions	The student is successfully enrolled to the battle.
Exceptions	If the registration deadline is expired, the system will throw an error message and the student will not be able to subscribe to the battle. The system will return to the list of battles.

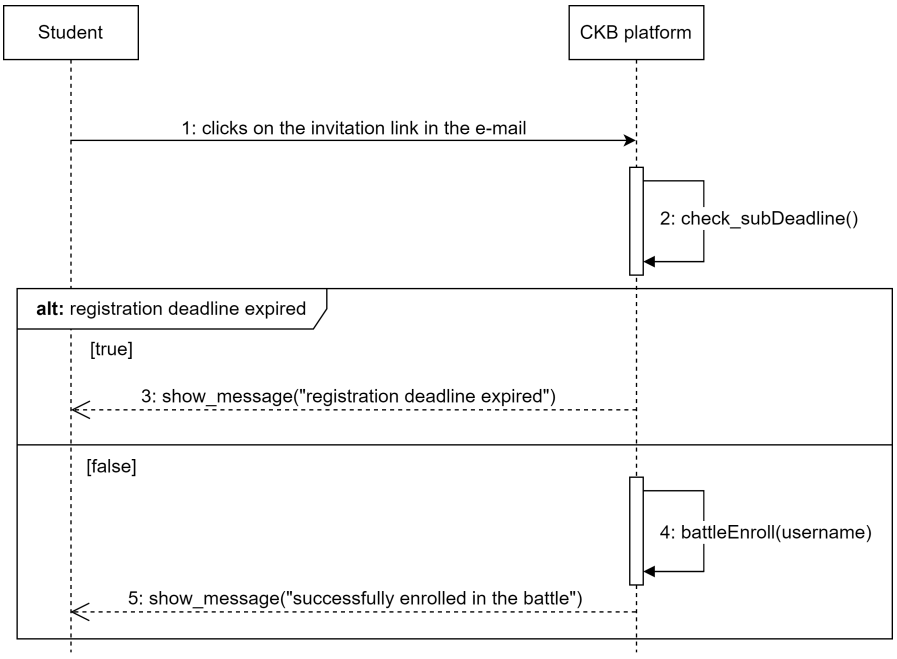

Figure 3.14: Enroll in a battle sequence diagram

UC.11 Invite a student

Name	Invite a student.
Actors	Students.
Entry conditions	A student, who is enrolled in a battle, wants to create a team.
Event flow	1. The student clicks on the button "my tournaments". 2. The student visualize the list of tournaments in which he/she is subscribed and all the battles in which he/she is enrolled. 3. The student selects a battle. 4. The student visualizes his/hers current team. 5. The student inserts the username of the student who he/she wants to invite. 6. The student clicks on the button "invite a student". 7. The system checks if the username inserted is enrolled in the battle.
Exit conditions	The invited student has received an e-mail.
Exceptions	• If the registration deadline for the battle is expired, the invite will not be sent. • If the invited student is not enrolled in the invite will not be sent. In both cases the system will throw an error message and return to the entry condition.

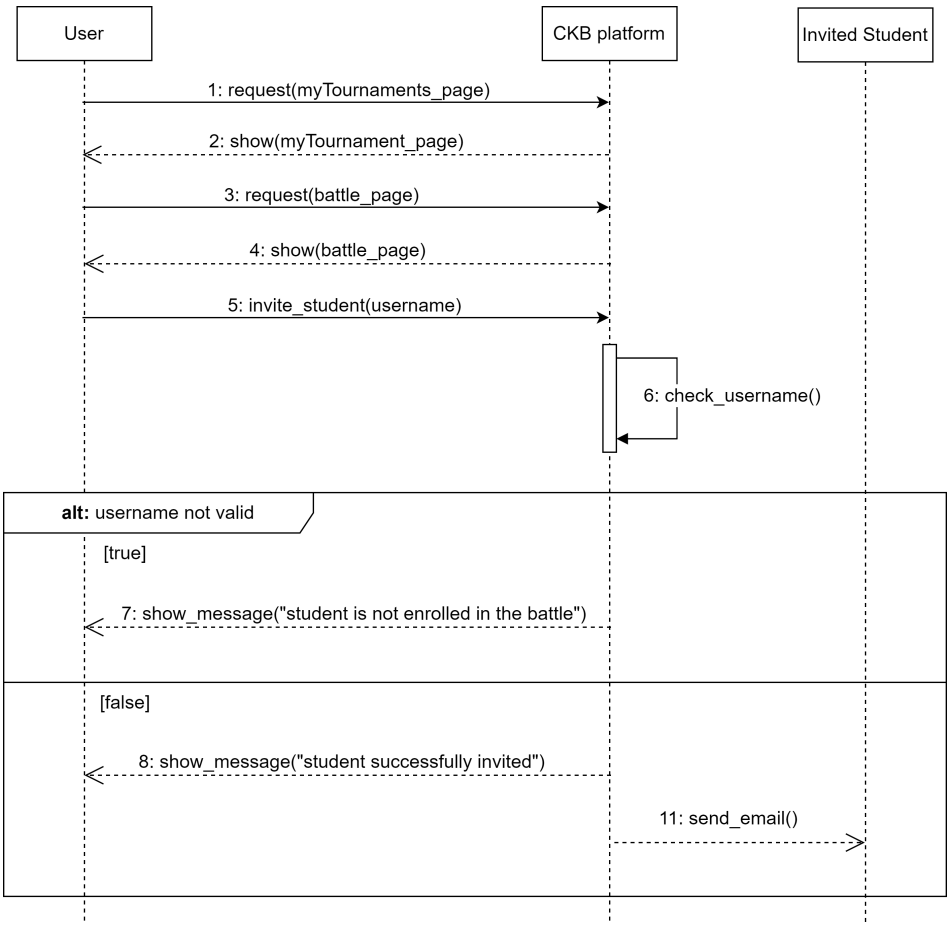

Figure 3.15: Invite students sequence diagram

UC.12 Student accepts invite

Name	Student accept an invite.
Actors	Students.
Entry conditions	A student receives an invitation by email and he/she wants to join the team.
Event flow	1. The student clicks on the link received by email. 2. The system check if the registration deadline is not expired. 3. The system check if the team is not complete.
Exit conditions	The student has successfully joined the team.
Exceptions	If the registration deadline is already expired or the team is already full the system will throw an error message and the student will not be able to leave the team. The system will return to the entry condition.

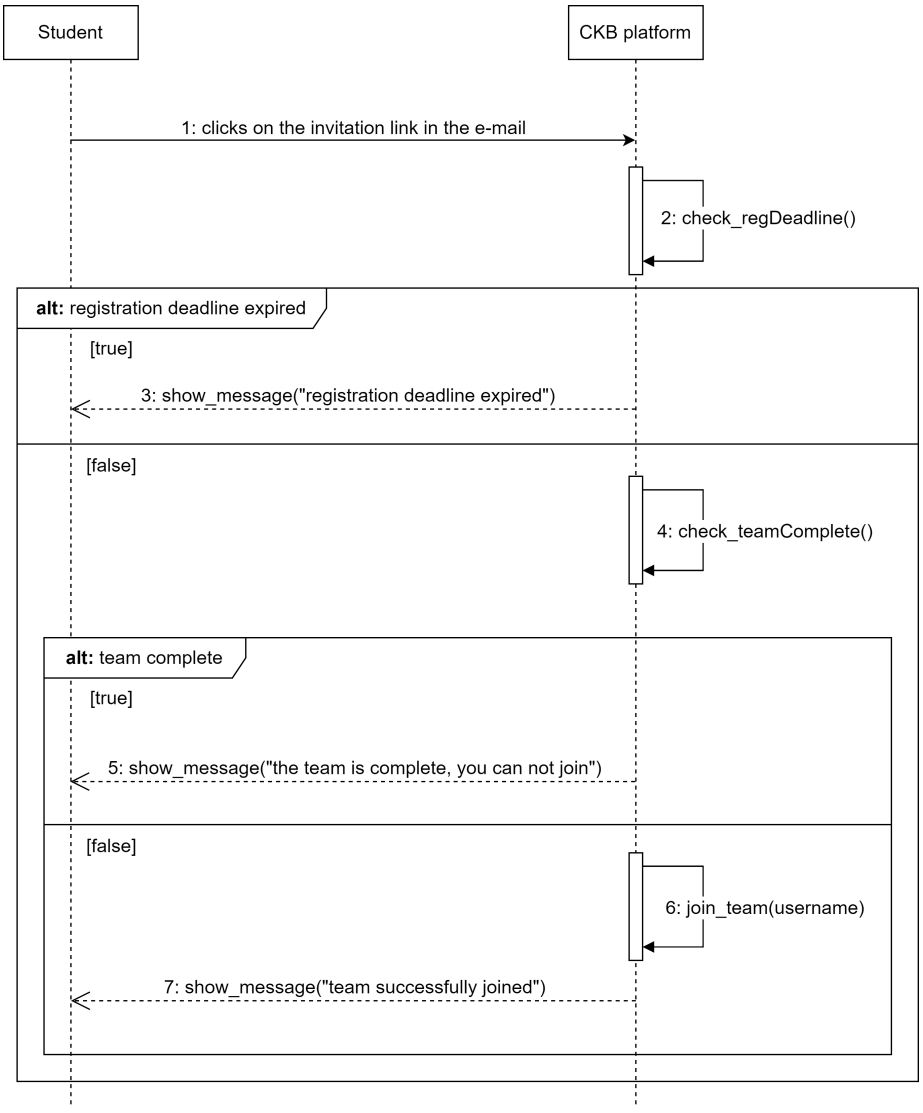

Figure 3.16: Student accepts invite sequence diagram

UC.13 Leave a team

Name	Leave a team.
Actors	Students.
Entry conditions	A student wants to leave his/hers current team.
Event flow	1. The student clicks on the button "my battles". 2. The student visualizes the list of battles he/she is registered to. 3. The student selects a battle. 4. The student visualizes his/hers current team. 5. The student clicks on the button "leave team".
Exit conditions	The student has successfully left his/her team.
Exceptions	If the registration deadline is already expired the system will throw an error message and the student will not be able to leave the team. The system will return to the entry condition.

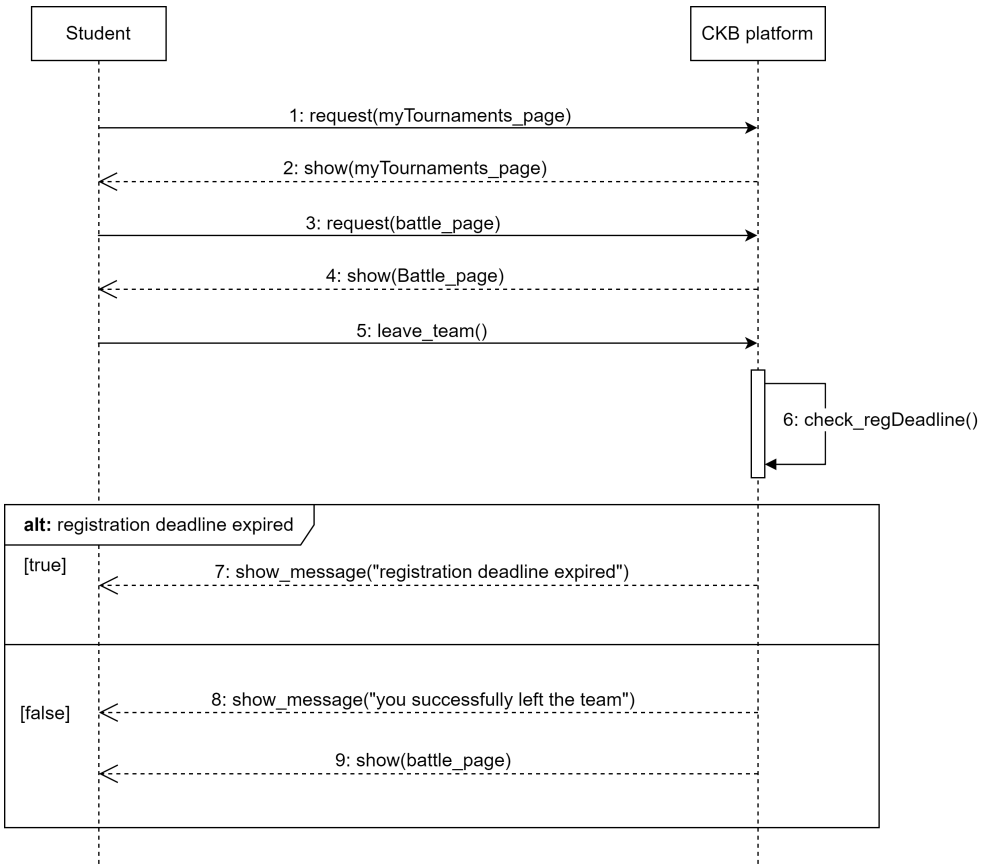

Figure 3.17: Student leaves team sequence diagram

UC.14 View battle ranking

Name	View battle ranking.
Actors	Educators and students.
Entry conditions	An user wants to see the real-time ranking of a battle.
Event flow	1. The user clicks on the button "my battles". 2. The user visualizes the list of battles that are correlated to him/her. 3. The user selects a battle. 4. The system show the real-time raking of the selected battle.
Exit conditions	The user visualize the real-time ranking of a battle.

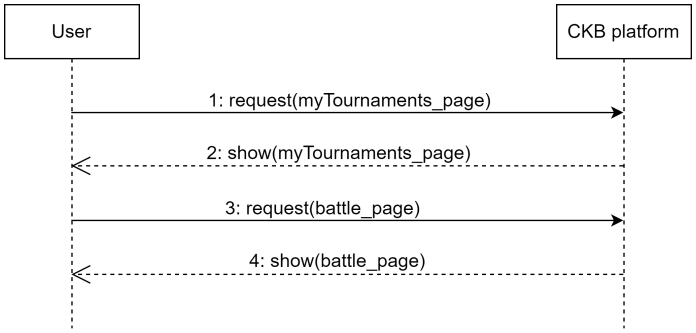

Figure 3.18: View battle ranking sequence diagram

UC.15 Perform manual evaluation

Name	Perform manual evaluation.
Actors	Educators.
Entry conditions	An educator wants to manually evaluate some students.
Event flow	1. The educator clicks on the button "my battles". 2. The educator visualize the list of battles that are correlated to him/her. 3. The educator select the battle he/she wants to evaluate. 4. The educator clicks on the button "perform manual evaluation". 5. The educator assigns a personal score to each participant in the battle. 6. The system checks if the rating exceeds the maximum score and truncates it if so. 7. The educator clicks on the button "end consolidation stage".
Exit conditions	The educator end the consolidation stage.

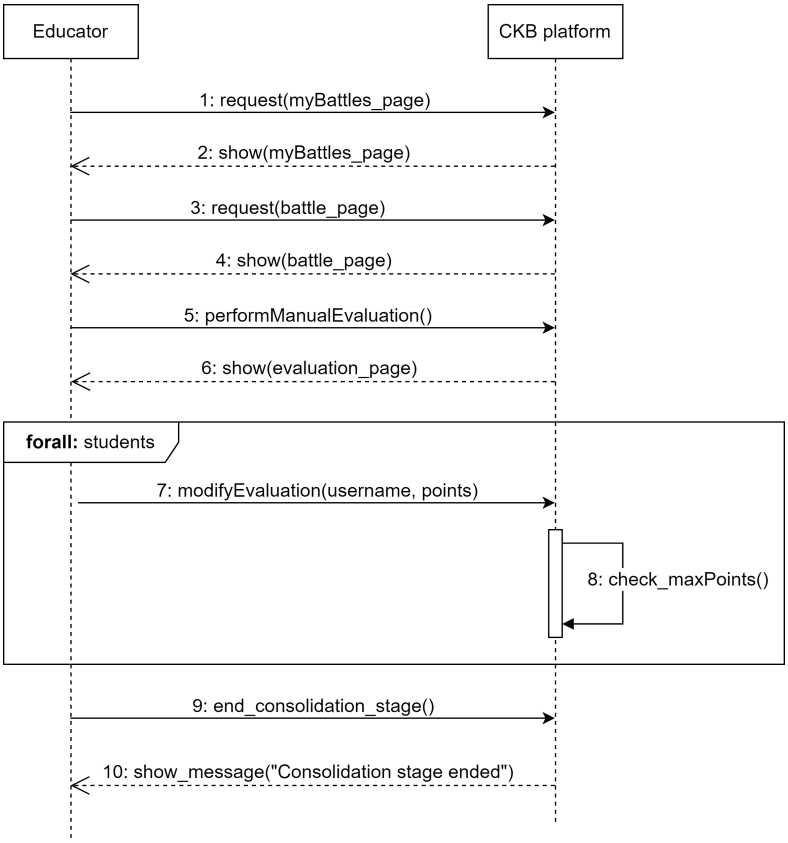

Figure 3.19: Perform manual evaluation sequence diagram

UC.16 Perform automatic evaluation

Name	Perform automatic evaluation.
Actors	GitHub API, static analysis tool.
Entry conditions	A team push a new solution on the GitHub repository.
Event flow	1. The GitHub API triggers the CKB platform. 2. The system request the quality report based on the parameter chosen at battle creation. 3. The static analysis tool gives back the quality report. 4. The system performs functional evaluation based on tests cases. 5. The system performs time evaluation. 6. The system updates the real-time battle ranking and the student's tournament ranking.
Exit conditions	The real-time rank and the student tournament ranking is updated.

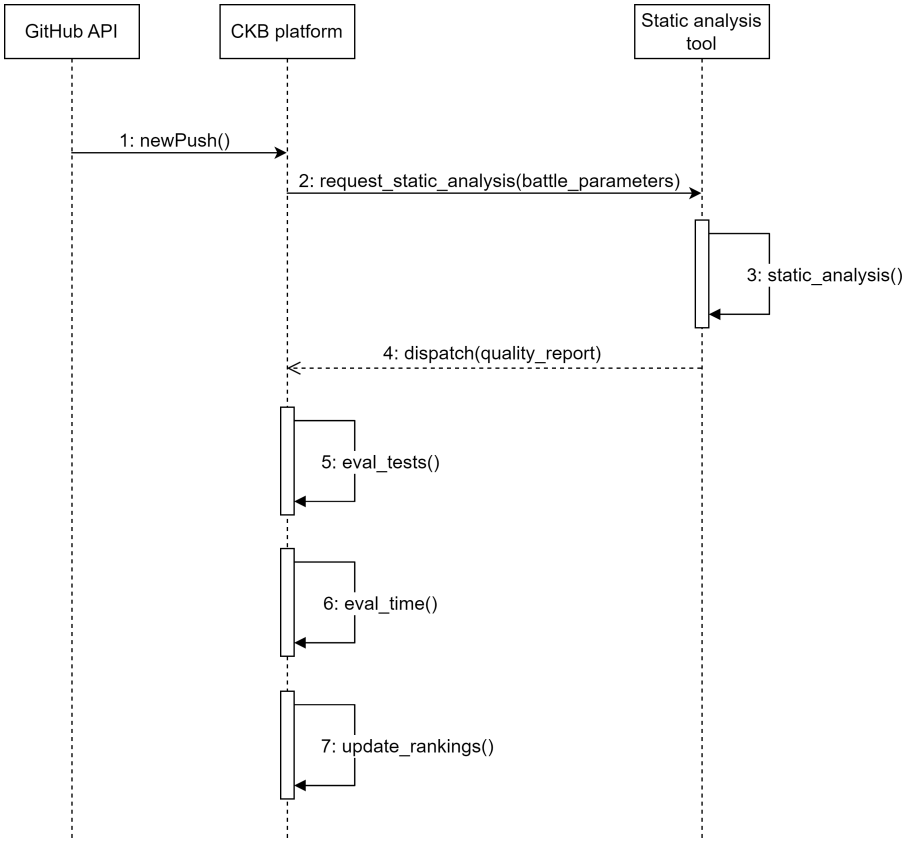

Figure 3.20: Perform automatic evaluation sequence diagram

UC.17 Create a badge

Name	Create a badge.
Actors	Educators.
Entry conditions	An educator wants to create a badge within a tournament. The educator is creating the tournament.
Event flow	1. The educator clicks on the button "create badge". 2. The educator inserts all information requested in a form. 3. The system check the correctness of the inserted information.
Exit conditions	The badge has been successfully created.

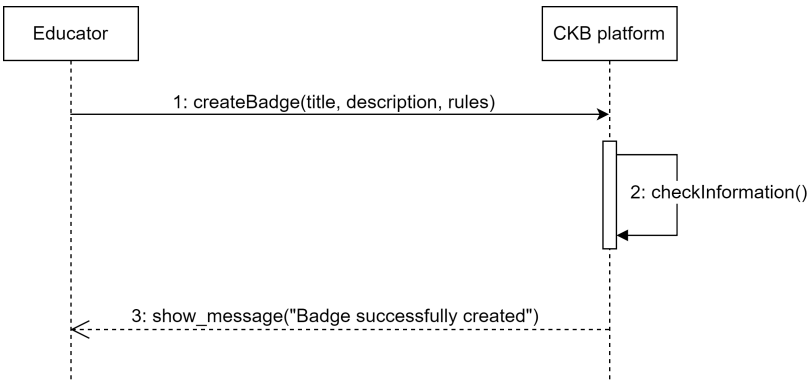

Figure 3.21: Create badge sequence diagram

UC.18 See student profile

Name	See student profile.
Actors	Educators and students.
Entry conditions	An user wants to see a personal profile of a student.
Event flow	1. The user inserts the username of the student he/she wants to see. 2. The user clicks on the button "see student profile". 3. The system check the inserted information. 4. The system show the personal profile requested if present.
Exit conditions	The user see the personal profile of a student.
Exceptions	If the selected username does not exist the system will throw an error message. The system will return to the entry conditions

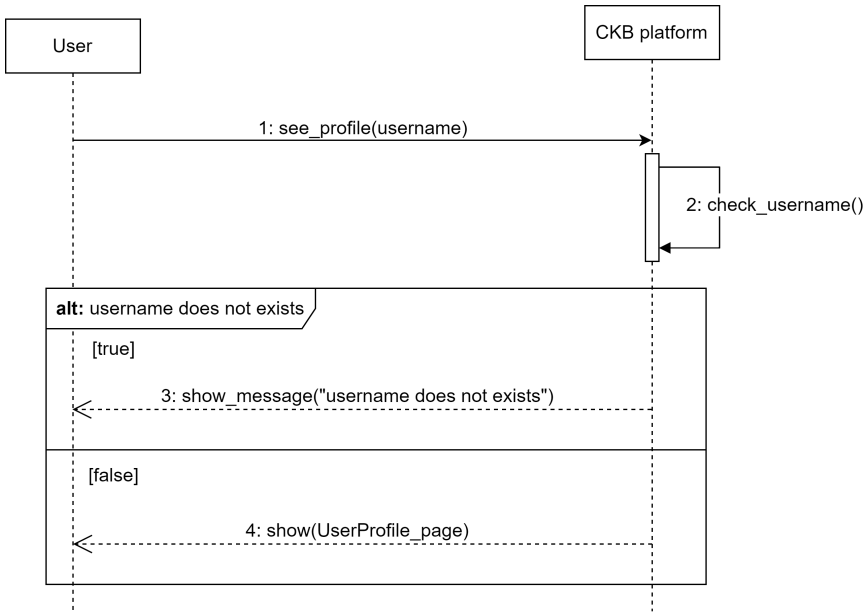

Figure 3.22: See student profile sequence diagram

3.2.4 Traceability matrix

Use Case ID	Goal ID	Req ID	Scenario
UC.1	ALL	R.1	SC.1
UC.2	ALL	R.2	SC.1
UC.3	G.1	R.7 R.8 R.16	SC.2
UC.4	G.3	R.9 R.17	SC.2
UC.5	G.3	R.8	SC.2
UC.6	G.3	R.8	SC.2
UC.7	G.1	R.8	SC.2
UC.8	G.8 G.9	R.3 R.16 R.5 R.21	SC.5
UC.9	G.4	R.3 R.13 R.16	SC.2
UC.10	G.5	R.14 R.15 R.17 R.19	SC.3
UC.11	G.6	R.15 R.18	SC.3
UC.12	G.6	R.15	SC.3

Use Case ID	Goal ID	Req ID	Scenarios
UC.13	G.6	R.15 R.18	SC.3
UC.14	G.7	R.4 R.5 R.10 R.20	SC.4
UC.15	G.7	R.10 R.20	SC.5
UC.16	G.7	R.4 R.5 R.20	SC.4
UC.17	G.2	R.11 R.12	SC.6
UC.18	G.10	R.6	SC.7

3.3 Performance Requirements

In this section are listed some performance requirements for the CKB platform that are essential to the efficiency of the entire system. The CKB platform should be able to guarantee the connection of 100.000 users simultaneously.
In less than 2 seconds, it should be able to respond to user interactions, such as page loading.
In less than 5 seconds, it should be able to send a response to a query and run its algorithms on the metadata.

3.4 Design Constraints

3.4.1 Standards Compliance

All users information will be processed in according with the General Data Protection Regulation (GDPR), and e-mail addresses won't be used for commercial purposes.
The system has to adopts international standards about date and time representation.
CKB system should use stateless protocols and standard operations to allow components to be managed and updated without affecting the system as a whole. It's crucial to design modules properly so that ease of use, security and performance will remain the core factors of the system.

3.4.2 Hardware Limitations

The CKB platform requires the users a personal device that is able to connect to the internet.

3.4.3 Any other constraint

The system will store all the data submitted in a standardize form, to make easier to catalog and run queries on it.
The database system supporting the CKB platform should be capable of handling efficiently great amount of data thought the use of proper indexing, query optimization, and database scaling strategies.

3.5 Software System Attributes

3.5.1 Reliability

The CKB platform should be highly reliable in order to guarantee the continuity of the service, every user should be able to access the platform anytime. The system should implements robust errors handling and fault tolerance mechanism to prevent error propagation and data loss.

3.5.2 Availability

The CKB platform should be available to users 24/7, without frequent interruptions. Since the system is not emergency-related, it should be up 99% of the time. This

means that the average downtime is around at 3.65 days per year.
In order to achieve this level of availability the system should implement a disaster recovery plan, monitoring and alerting tools, and the ability to react fast to resolve any system issues.

3.5.3 Security

As the system store some personal information about the users, security is an important issue. This stored data must be encrypted, and passwords must also be hashed. Every time a password need to be recovered a new one must be created, by sending a verification e-mail that contain a time-limited that confirm the user identity and gives them access to reset their password.

3.5.4 Maintainability

The CKB system should be divided in different modules implementing the various functionalities. In this way ordinary maintenance and/or future fixes or improvements will be easier to be performed.
Maintenance and updates must be scheduled in advance so that they do not interfere with ongoing battles.

3.5.5 Portability

The CKB web platform should be accessible from any web browser.

3.5.6 Usability

The user interfaces of the platform should be easy to use and intuitive. It is expected that the 99% of users will be able to use the platform without assistance.

Chapter 4

Formal Analysis Using Alloy

4.1 Alloy Model

In this section is presented a formal model of the CKB platform using alloy. The model is a simplification of the entire system and represent only the most significant parts.

```
//defining date
sig Date {}

//defining users
sig Username {}
abstract sig User{
  username :  Username,
}
sig Educator extends User{}
sig Student extends User{
  collectedBadges : set Badge
}

//defining team structure
sig TeamId{}
sig Team{
      teamId: TeamId,
  battleId : BattleId,
  members : set Student,
  size:Int
}
{
  #members <= size
}

//defining tournament and tournament rankings
sig TournamentId{}
abstract  sig State{}
one sig Open extends State {}
one sig Close extends State{}
one sig Ongoing extends State{}
sig Points{}
```

```
sig TournamentScore{
  student : Student,
  tournamentPoints : Points
}
sig Tournament {
      tournamentId : TournamentId,
  creator : one Educator,
  collaborators : set Educator,
  battles : set Battle,
  participants : set Student,
  subscriptionDeadline : Date,
  badges : set Badge,
  state : one State,
  tournamentRanking : set TournamentScore
}

//defining battle and battle rankings
sig BattleScore{
  team : Team,
  battlePoints : Points
}
sig CodeKata{}
sig BattleId{}
sig Battle{
      battleId : BattleId,
      tournamentId : TournamentId,
  creator : Educator,
  code : one CodeKata,
  participants : set Student,
  teams : set Team,
  maxTeamSize : Int,
  minTeamSize : Int,
  registrationDeadline: Date,
  submissionDeadline: Date,
  state : one State,
  battleRanking : set BattleScore
}
{
  maxTeamSize <=4
  minTeamSize > 0
  maxTeamSize >= minTeamSize
}

//defining badge structure
sig BadgeId{}
sig Badge {
  tournamentId : TournamentId,
  badgeId : BadgeId,
}

-----------------------------------------------------
--Facts
-----------------------------------------------------
```

```
86
87  //Tournament
88
89  //ensures that the tournamentID is unique
90  fact uniqueTournamentId{
91        all t1,t2 : Tournament | t1!=t2 => t1.tournamentId != t2.
        tournamentId
92  }
93
94  //ensures that the tournamentID exists only if exists a
        tournament with this id
95  fact tournamentIdHasTournament{
96    all ti : TournamentId | one t:Tournament | t.tournamentId =
        ti
97  }
98
99  //ensures that the creator of the tournament is not in the
        collaborators list
100 fact creatorIsNotCollaborator{
101       all t : Tournament | t.creator not in t.collaborators
102 }
103
104 //ensures that all the battles in a tournament have the right
        tournament id
105 fact allBattlesHaveOneTournamentId{
106       all t : Tournament, b:t.battles| b.tournamentId = t.
        tournamentId
107 }
108
109 //Battle
110
111 //ensures that a battle exists only in a tournament
112 fact battleExistsOnlyInATournament{
113   all b : Battle | one t:Tournament | b in t.battles
114 }
115
116 //ensures that the battleID is unique
117 fact uniqueBattleId{
118       all b1,b2 : Battle | b1 != b2 => b1.battleId != b2.
        battleId
119 }
120
121 //ensures that the battleID exists only if exists a battle with
        this id
122 fact noBattleIdWithoutBattle{
123   all bi : BattleId | one b:Battle | bi in b.battleId
124 }
125
126 //ensures that the creator of a battle is part of the
        tournament
127 fact battleCreatorIsInTournament{
128   all t : Tournament, b : t.battles | (b.creator in t.
        collaborators) or ( b.creator=t.creator)
```

```
129 }
130
131 //ensures that if a student is enrolled in a battle then he/she
        is a participant of the tournament
132 fact ifStudentInBattleThenInTournament{
133   all t : Tournament, b : t.battles, s : Student| s in b.
        participants => s in t.participants
134 }
135
136 //ensures that a CodeKata exists only if it is the code of a
        battle
137 fact noCodekataWithoutBattle{
138   all ck : CodeKata | one b : Battle | ck = b.code
139 }
140
141 //User
142
143 //ensures that all the username are unique
144 fact usernameIsUnique{
145   all u1,u2 : User | u1 !=u2 => u1.username != u2.username
146 }
147
148 //ensures that a username exists only if it is the username of
        a user
149 fact NoUsernameWithoutUser{
150   all un : Username | one u : User | un = u.username
151 }
152
153 //Team
154
155 //ensures that a team exists only in a battle
156 fact teamExistsOnlyInOneBattle{
157   all t : Team | one b : Battle | t in b.teams
158 }
159
160 //ensures that a user cannot join different teams in the same
        battle
161 fact NoSharedPlayers {
162     all b : Battle, t1, t2: b.teams | t1!=t2 => ((t1.members &
        t2.members) = none)
163 }
164
165 //ensures that the teamID is unique
166 fact uniqueTeamId{
167       all b : Battle| all t1,t2 : b.teams | t1!=t2 => t1.teamId
        != t2.teamId
168 }
169
170 //ensures that a TeamId exists only if it is the id of an
        existing team
171 fact noTeamIdWithoutTeam{
172   all ti : TeamId | one t:Team | ti in t.teamId
173 }
```

```
174
175 //ensures that all the teams of a battle have the right
    battleID
176 fact allTeamsHaveTheRightBattleId{
177       all b : Battle, t : b.teams | t.battleId = b.battleId
178 }
179
180 //ensures that a team is composed of at least one student
181 fact noTeamWithoutStudents{
182   all tm: Team | some s: Student | s in tm.members
183 }
184
185 //ensures that all the students of a team are participants of
    the battle
186 fact allTeamStudentAreInBattle{
187   all b : Battle, s : b.teams.members | s in b.participants
188 }
189
190 //ensures that all the teams respect the limitations in size
191 fact allTeamsRespectMaxMin{
192   all b : Battle, t : b.teams | t.size <= b.maxTeamSize and t.
    size >= b.minTeamSize
193 }
194
195 //Badges
196
197 //ensures that a badge exists only if belongs to an existing
    tournament
198 fact badgeExistsOnlyInATournament{
199   all b : Badge | one t : Tournament | b in t.badges
200 }
201
202 //ensures that the badgeId is unique
203 fact uniqueBadgeId{
204     all b1,b2 : Badge | b1 != b2 => b1.badgeId != b2.badgeId
205 }
206
207 //ensures that a BadgeId exists only if it is the id of a Badge
208 fact noBadgeIdWithoutBadge{
209   all bi : BadgeId | one b : Badge | bi in b.badgeId
210 }
211
212 //ensures that a student, who is not a participant of a
    tournament, cannot collects the tournament's badges
213 fact StudentThatAreNotInATournamentCannotHaveItsBadges{
214   all s : Student, t : Tournament |
215     (s not in t.participants) implies not (one b : Badge | b in
    s.collectedBadges and b in t.badges)
216 }
217
218 //ensures that if a tournament is closed there is at least one
    participant that have earned its badges
219 fact ifATournamentIsClosedSomeOfItsStudentsHaveItsBadge{
```

```
220   all t : Tournament, bd : t.badges | t.state = Close => bd in
      t.participants.collectedBadges
221 }
222
223 //ensures that if a tournament is not closed there is not any
      participant that have earned its badges
224 fact ifATournamentIsNotClosedAllItsBadgesAreNotAssigned{
225   all t : Tournament, bd : t.badges | t.state != Close => bd
      not in t.participants.collectedBadges
226 }
227
228 //State
229
230 //ensures that if a tournament is in open there are not any
      battles in it
231 fact ifTournamentIsOpenDontContainsBattles{
232   all t : Tournament | t.state = Open => t.battles = none
233 }
234
235 //ensures that if a tournament is closed all the battles of
      this tournament are closed
236 fact ifTournamentIsCloseAllBattlesMustBeClose{
237   all t : Tournament, b : t.battles | t.state = Close => b.
      state = Close
238 }
239
240 //Scores and Rankings
241
242 //ensures that the cardinality of a tournament ranking is equal
       to the number of participants in it
243 fact CardinalityCheckForTScores{
244   all t : Tournament| #t.participants = #t.tournamentRanking
245 }
246
247 //ensures that the cardinality of a battle ranking is equal to
      the number of participants in it
248 fact CardinalityCheckForBScores{
249   all b : Battle | #b.teams = #b.battleRanking
250 }
251
252 //ensures that the ranking of a tournament is only composed by
      participants of the tournament
253 fact NoStudentEnrolledWithoutTScore{
254   all t : Tournament, s : t.participants | one tlt :t.
      tournamentRanking.student|s = tlt
255 }
256
257 //ensures that the ranking of a battle is only composed by
      teams of the battle
258 fact NoTeamWithOutBScore{
259   all b : Battle, t : b.teams |one blt : b.battleRanking.team|t
       =blt
260 }
```

```
261
262 //ensures that a tournament score exists only if it is in a
      tournament ranking
263 fact everyTScoreBelongsToT{
264   all ts : TournamentScore | one t : Tournament | ts in t.
      tournamentRanking
265 }
266
267 //ensures that a battle score exists only if it is in a battle
      ranking
268 fact everyBScoreBelongsToB{
269   all bs : BattleScore | one b : Battle | bs in b.battleRanking
270 }
271
272 -------------------------------------------------
273 --Predicates
274 -------------------------------------------------
275
276
277 //The rules described in the following predicates don't belong
      strictly to the model
278 //however are usefull to underline the not trivial solution to
      rapresent the model,
279 //in this way we can observe clearly the structure of the model
      .
280
281 //ensures that there isn't any student that is not a
      participant of a tournament
282 pred allStudentEnrolled{
283   all s : Student | s in Tournament.participants
284 }
285
286 //ensures that there isn't any educator that is not a
      participant of a tournament
287 pred allEducatorsInvolved{
288   all e : Educator | (e in Tournament.creator ) or (e in
      Tournament.collaborators)
289 }
290
291 //ensures that every student participates in at least a battle
292 pred NofreeStudentInTournament{
293   all t : Tournament, s : t.participants | one b: t.battles | s
       in b.participants
294 }
295
296 //ensures that every student participates in at least a team
297 pred NofreeStudentInABattle{
298   all b : Battle, s : b.participants | one t : b.teams | s in t
      .members
299 }
300
301
302 //WORLD GENERATION
```

```
303
304 //WORLD1: multi-tournament and multi-battle structure
305 pred world1 {
306   NofreeStudentInABattle
307   NofreeStudentInTournament
308   allEducatorsInvolved
309   allStudentEnrolled
310   #Tournament = 2
311   #Battle = 2
312   #Educator = 2
313   #Student = 4
314   #Badge = 0
315   all t:Tournament  | one b:Battle | b in t.battles
316   all b:Battle | some s:Student | s in b.participants
317
318 }
319 run world1 for 10
320
321 //WORLD2: score system and team structure
322 pred world2 {
323   NofreeStudentInABattle
324   NofreeStudentInTournament
325   allEducatorsInvolved
326   allStudentEnrolled
327   #Tournament = 1
328   #Battle = 1
329   #Educator = 1
330   #Student = 7
331   #Badge = 0
332   #Team > 2
333
334 }
335 run world2 for 10
336
337 //WORLD3: badge management and assignment due to tournament
    state
338 pred world3 {
339   NofreeStudentInABattle
340   NofreeStudentInTournament
341   allEducatorsInvolved
342   allStudentEnrolled
343   #Tournament = 2
344   #Battle = 1
345   #Educator = 1
346   #Student = 2
347   #Badge = 2
348   one t:Tournament | t.state = Close
349   one t:Tournament | t.state != Close
350   all t:Tournament | one b:Badge | b in t.badges
351
352 }
353 run world3 for 10
```

World 1

In world 1 (figure 4.1) are represented the interactions between multiple tournaments, multiple battles, and some educator and students.

World 2

In world 2 (figure 4.2) is represented the score system (with rankings) and the team structure of a battle.

World 3

In world 3 (figure 4.3) is represented the badge management and its assignment due to the tournament state.

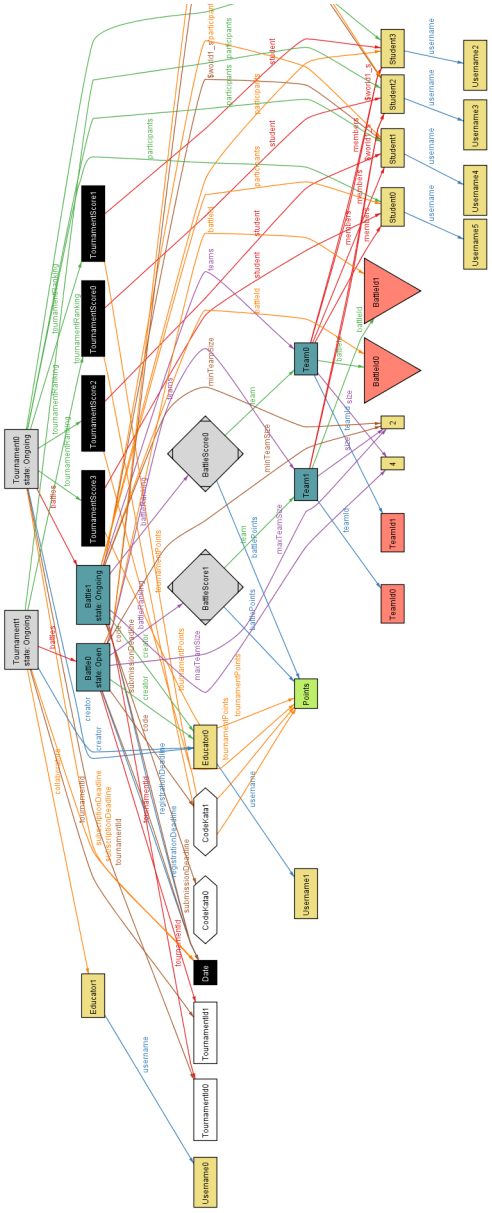

Figure 4.1: World 1

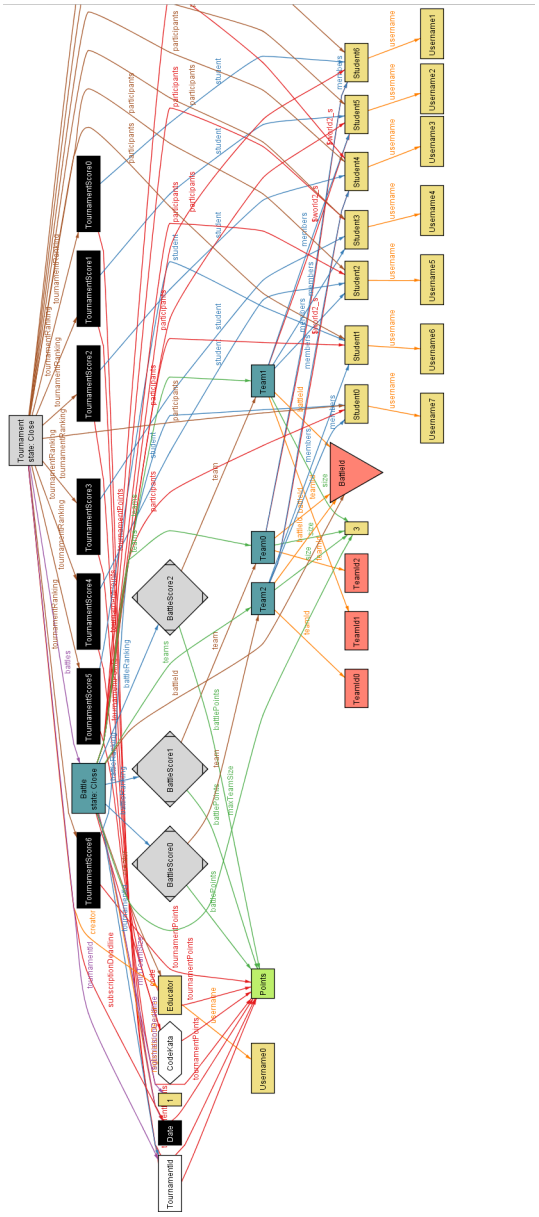

Figure 4.2: World 2

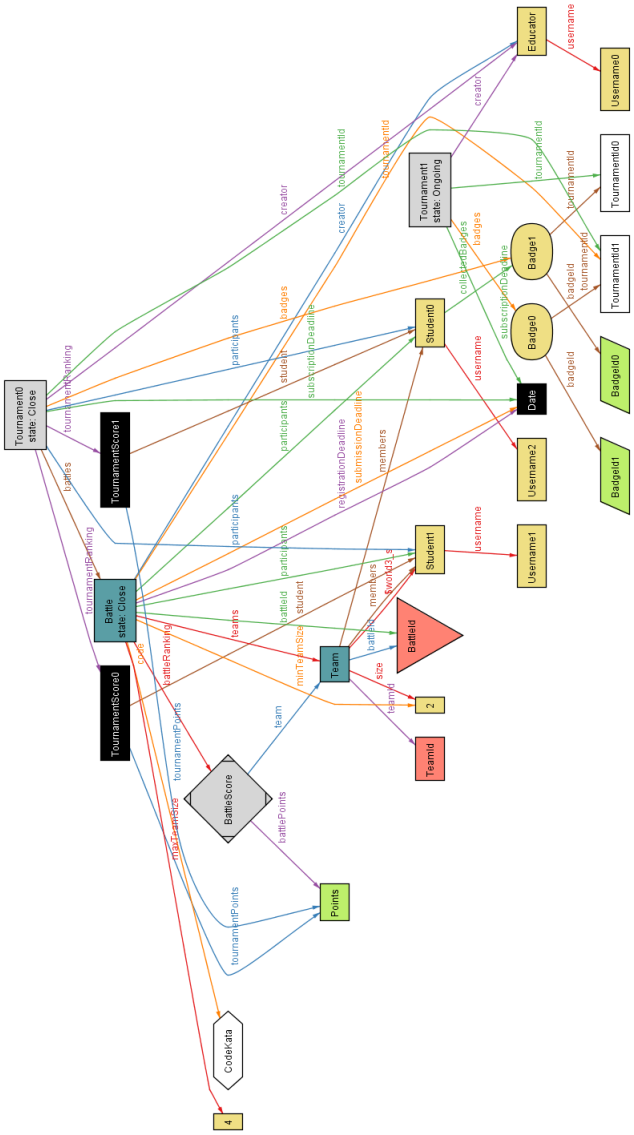

Figure 4.3: World 3

Chapter 5

Effort Spent

5.1 Teamwork

Task	Hours
Initial briefing	2
Alloy	1

5.2 Eliahu Itamar Cohen

Task	Hours
Chapter 1: Introduction	0
Chapter 2: Overall description	1.5
Chapter 3: Specific requirements	5.5
Chapter 4: Alloy	4.5
Extra Activities	2
Total	**13.5**

5.3 Marco Gervatini

Task	Hours
Chapter 1: Introduction	0
Chapter 2: Overall description	2
Chapter 3: Specific requirements	7
Chapter 4: Alloy	12.5
Extra Activities	3.5
Total	**25**

5.4 Caterina Motti

Task	Hours
Document setup	2.5
Chapter 1: Introduction	3
Chapter 2: Overall description	16
Chapter 3: Specific requirements	22
Chapter 4: Alloy	1
Extra Activities	4
Review	3.5
Total	**52**

Chapter 6

References

- All diagrams have been made with https://www.draw.io
- Alloy code development has been supported by VS code (Extensions used: Alloy, Alloy VSCode) and Alloy Analyzer